WEIGHT SPACE LEARNING WITHOUT WEIGHTS: LEVERAGING NEURAL OPERATORS FOR MODEL EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks increasingly serve as representations of continuous signals, from images to dynamical systems. Editing these representations raises a question: should transformations act on discretized signals, network parameters, or the functions they encode? Weight-space learning operates directly on parameters, but neural representations are non-unique, while methods operating on discretized signals can depend strongly on grid resolution. Here, we formulate editing as an operator learning problem and compare our neural operator-based approach to alternatives that operate on discretized signals or weights, using rotating MNIST implicit neural representations and introducing damping into harmonic oscillator neural ODEs as examples. For fixed-angle rotation, our approach, instantiated using Fourier neural operators, performs competitively on the original grid and outperforms weight-space baselines and vision models at intermediate coordinates. Furthermore, angle-conditioned operators generalize to held-out rotations. For oscillator damping, our neural operator-based approach outperforms weight space approaches. These results support model editing as a practical approach to transforming neural representations, with advantages in generalization across sampling grids and families of transformations.

1 INTRODUCTION

Many signals of interest (e.g., physical fields or spatial trajectories) are naturally continuous but are typically represented and processed using a finite number of samples or expansions into a set of fixed basis functions. More recently, parameterizing such signals as neural networks has emerged as a more faithful and resolution-independent alternative (Xie et al., 2022; Dupont et al., 2022a). Implicit Neural Representations (INRs) (Sitzmann et al., 2020) and Neural Radiance Fields (NeRFs) (Mildenhall et al., 2021) are prominent examples of this approach and have been applied, e.g., to images, radiance fields, and videos. Similarly, neural ordinary differential equations (neural ODEs) and their variants are neural representations of a differential equation's vector field (Chen et al., 2018; Kidger, 2022). More broadly, this approach encompasses implicit behavioral cloning (Florence et al., 2022) in imitation learning and deep generative models as neural representations of probability distributions (Tomczak, 2022).

In many applications, however, continuous signals must not only be stored but also modified. For example, weather prediction requires modifying neural representations of physical fields as time evolves or as new measurements become available. Similarly, autonomous vehicles must be able to dynamically update neural representations of the 3D environments they navigate. Such changes to neural representations of continuous signals are a model editing task, and approaching it using machine learning requires answering a basic question: should learned transformations operate on model weights, discretized signals, or the encoded function?

The emerging field of weight space learning (WSL) treats the parameters of trained neural networks as structured objects that can themselves be analyzed, represented, and generated (Han et al., 2026). This perspective is increasingly useful as trained neural networks themselves become reusable data artifacts, rather than just tools for prediction. However, tackling model editing directly in weight space is challenging because neural network parameterizations are non-unique: permutations and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

other symmetries can produce distinct sets of parameters representing the same function (Han et al., 2026; Navon et al., 2023). This redundancy complicates the geometry of weight space and can impair generalization when equivalent parameterizations are not well represented in the training data (Shamsian et al., 2024). Existing approaches address the symmetries through canonicalization (Ainsworth et al., 2023; Gupta et al., 2026), invariant or equivariant architectures (Bronstein et al., 2021; Navon et al., 2023), data augmentation (Shamsian et al., 2024), or invariant or functional representations (Zhou et al., 2023; Kahana et al., 2025; Herrmann et al., 2024).

While alternative approaches, such as learning transformations from discretized signals, avoid weight-space ambiguity, they typically tie the learned transformation to a particular discretization of an underlying continuous signal, thus sacrificing many of the benefits of neural representations. Neural operators (Kovachki et al., 2023; Li et al., 2021), learnable mappings between function spaces, can be evaluated across discretizations and thus overcome this limitation.

Here we propose using neural operators (Kovachki et al., 2023; Li et al., 2021) for editing neural representations directly in the space of continuous signals. This has the advantage that functionally equivalent parameterizations collapse to the same represented function. Moreover, the learned transformation is not inherently tied to a fixed sampling grid. We formalize this approach mathematically and illustrate it through experiments on INRs and neural ODEs.

We make four main **contributions**:

- We formalize model editing for neural representations of continuous signals as an operator learning problem.
- We show that rotating INRs using neural operators preserves resolution independence of neural representations better than approaches based on weights or discretized signals, while performing competitively when evaluated at the resolution of the INR training data.
- We demonstrate that neural operators can be trained to rotate INRs by arbitrary angles without the need to generate INRs of rotated images, as required for weight space approaches.
- We provide evidence that neural operators can edit neural ODEs trained on harmonic oscillators to include damping, outperforming baselines operating in weight space.

2 Related Work

Representing and editing continuous signals Neural networks are advantageous as representations of continuous signals as they can be queried at arbitrary points of the underlying domain and easily differentiated with respect to both weights and coordinates, making it possible to optimize the stored signals (Xie et al., 2022; Dupont et al., 2022a). This approach was especially influential in computer vision, where it has been applied to images (Sitzmann et al., 2020), object geometries (Park et al., 2019), and scenes (Mildenhall et al., 2021). In addition, several scientific machine learning techniques can be viewed through this lens: neural ODEs (Chen et al., 2018; Kidger, 2022) are encodings of a differential equation's vector field, and physics-informed neural networks (PINNs) (Raissi et al., 2019; Cuomo et al., 2022) store the solution of a (partial) differential equation. There are several approaches to modifying the stored signals, but they differ in important aspects from our operator-based approach. Conditioning on latent variables can be incorporated during training (Jang & Agapito, 2021; Hagnberger et al., 2024), but this requires anticipating the transformations a priori, whereas we propose a post-processing technique. Optimization-based approaches can introduce specific modifications post-hoc (Liu et al., 2021), while we aim to learn an operator that generalizes over families of transformations. Finally, generative models for neural representations synthesize new models and, thus, serve a different goal than our editing approach (Dupont et al., 2022b; Erkoç et al., 2023; Chou et al., 2023; Du et al., 2024).

Weight Space Learning Recent work in weight space learning spans model analysis, representation learning, and parameter generation (Han et al., 2026). A major line of work develops architectures that operate directly on neural network parameters while respecting their structure. Deep Weight Space Networks introduce permutation-equivariant layers for processing collections of network weights, with applications including INR classification (Navon et al., 2023). Neural Functional Transformers also support editing of INR weights to transform the represented signals (Zhou et al., 2023). Related work extends this idea through attention-based neural functionals, graph-based

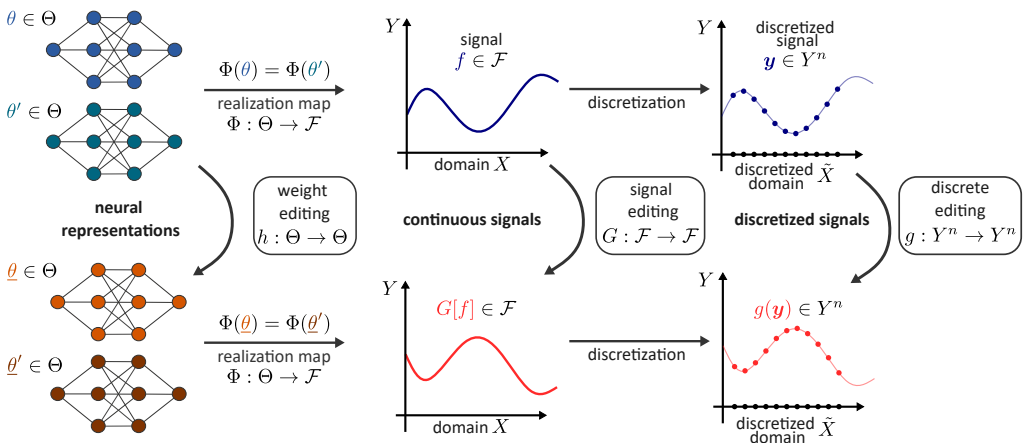

Figure 1: A *neural representation* f_θ of a *continuous signal* $f \in \mathcal{F}$ on a domain X encodes the signal into the weights $\theta \in \Theta$ of a neural network. While this representation is resolution independent, it is not unique, i.e., there are weights $\theta' \neq \theta$ encoding the same signal. Representing the signal as a *discretized signal* avoids non-uniqueness but is tied to a specific discrete domain $\tilde{X}$ and thus resolution. Changes to a continuous signal can be expressed as an operator $G : \mathcal{F} \to \mathcal{F}$. We propose to view editing of neural representations as an operator learning problem, i.e., to learn G using a neural operator G_ω. In contrast to weight editing, i.e., learning a function $h : \Theta \to \Theta$, this approach does not suffer from non-uniqueness of the learning target, and, in contrast to learning a function $g : Y^n \to Y^n$ between discretized signals, it preserves resolution independence.

metanetworks, and architectures designed to accommodate broader classes of network structures and symmetries (Zhou et al., 2023; Lim et al., 2024; Zhou et al., 2024). Other approaches modify the representation of the network before downstream learning. Git Re-Basin aligns independently trained models through neuron permutations (Ainsworth et al., 2023), while more recent generative methods similarly canonicalize parameterizations before modeling them (Gupta et al., 2026). Alternatively, Neural Functional Transformers construct permutation-invariant representations of INRs (Zhou et al., 2023). While some weight space approaches construct invariant representations from weights, we start from data that is already invariant under weight transformations—the function encoded in the weights.

Neural Operators Neural operators provide a general framework for learning maps between function spaces (Kovachki et al., 2023). Much of the literature has focused on scientific machine learning, particularly modeling parametric partial differential equations and other physical systems. DeepONet learns nonlinear operators using separate branch and trunk networks for representing input functions and evaluation coordinates (Lu et al., 2021), while the Fourier neural operator performs nonlocal transformations through learned Fourier modes (Li et al., 2021). Spatial-domain neural operators instead parameterize the underlying integral kernel directly (Kovachki et al., 2023). A motivation for these approaches is that the target mapping is formulated at the level of functions rather than a particular numerical representation, enabling models designed to generalize across different discretizations. Subsequent work has developed operator-learning methods for a wide range of PDEs and physical systems, but comparatively little work has considered neural operators as tools for transforming the functions encoded by trained neural representations (El Hadramy et al., 2026; Pal et al., 2024). We study this setting directly, applying both Fourier and spatial-domain neural operators to transformations of INRs and comparing them with conventional image and parameter space approaches.

3 Model Editing as an Operator Learning Problem

We study the question of editing neural representations of continuous signals from the perspective of operator learning. In this section, we formalize this approach, which works in function space, and

contrast it with approaches that process signals on discretized domains or in weight space. Figure 1 provides an overview of these different approaches.

3.1 Representing continuous signals

Let $f : X \to Y$ be a *continuous signal* on the domain X and taking values in a set Y. We denote the space of such signals as $\mathcal{F}$ and emphasize that it is infinite dimensional. For example, X and Y could be the real numbers $\mathbb{R}$, and $f(x) \in Y$ describe the position of a particle at time $x \in X$.

A *neural representation* of the signal f is a neural network $f_\theta : X \to Y$ with weights $\theta \in \Theta$ such that, ideally, $f(x) = f_\theta(x)$ for all $x \in X$. As Θ is finite dimensional for conventional neural networks, neural representations are finite dimensional. Neural representations have the advantage that they can be evaluated at arbitrary coordinates $x \in X$, i.e., are not tied to a fixed resolution. They can be seen as expansions into basis functions that are learned from data and, thus, tailored to the application domain. Neural representations are, however, not unique. Due to weight space symmetries, one can thus find weights $\theta' \neq \theta$ such that $f_{\theta'}(x) = f_\theta(x)$ for all $x \in X$. This means the *realization map* $\Phi : \Theta \to \mathcal{F}$ is not injective.

An alternative finite-dimensional representation of the signal f relies on discretizing the domain X to obtain the discretized domain $\tilde{X}$. Evaluating f at n points $x_1, \ldots, x_n$, one can represent the signal as $\boldsymbol{y} = (f(x_1), \ldots, f(x_n)) \in Y^n$. We call this representation the *discretized signal*. In contrast to neural representations, this representation of the signal is tied to a fixed resolution, set by the number of samples n.

3.2 Modifying continuous signals

While neural representations have many desirable properties, they usually encode a fixed signal, raising the question of how the stored signal can be modified. One might want to modify a neural representation to adapt it according to changes in the physical system generating the data, e.g., update a neural ODE trained on past epidemiological data to the properties of a new viral strain, or one might want to change the stored signal, e.g., to remove inflammatory imagery from pictures stored as INRs.

Such changes to neural representations are *model editing* tasks. We propose to view model editing as an operator learning task. Given an operator $G : \mathcal{F} \to \mathcal{F}$ that takes the original signal f (or its neural representation f_θ) to the transformed signal $G[f]$ (or $G[f_\theta]$, respectively), we propose to learn G using a neural operator $G_\omega : \mathcal{F} \to \mathcal{F}$, where $\omega \in \Omega$ are the neural operator's weights. The benefits of this approach are that it preserves the resolution independence of the neural representation, circumvents non-identifiability issues that arise in weight space, and does not require access to neural representations of the transformed signal.

Following (Kovachki et al., 2023), a neural operator processes functions through layers that combine pointwise transformations with information aggregated across the domain. The input is first lifted to a multichannel feature function $v_0 = P(f)$, updated through layers of the form:

$$v_{\ell+1} = \sigma\left(W_\ell v_\ell(x) + b_\ell(x) + \int_D \kappa_\ell(x, y) v_\ell(y) dy\right), \qquad G_\omega[f] = Q(v_\ell) \tag{1}$$

where P, Q are learned pointwise maps and κ_ℓ is a learned kernel. In practice, functions are represented by sampled values and integrals are approximated numerically. The learned parameters are shared across discretizations.

We contrast our approach with weight space approaches and approaches relying on discretized signals. The weight space approach to model editing relies on neural representations $f_{\underline{\theta}}$ of the transformed signal and aims to learn a function $h : \Theta \to \Theta$ that takes the weights of a neural representation of the original signal to the weights of a neural representation of the transformed signal. Note that, as different weights can encode the same signal, such a function h is not uniquely defined by the operator G, leading to an identifiability problem in the weight space approach to model editing. In Fig. 1, both θ and $\theta' \neq \theta$ encode f, and both $\underline{\theta}$ and $\underline{\theta}' \neq \underline{\theta}$ encode $G[f]$, giving rise to $2^2 = 4$ possible functions h.

Finally, if the signal f is stored as a discrete signal $\boldsymbol{y} \in Y^n$, then one can attempt to learn a function $g : Y^n \to Y^n$ that maps the discrete signal $\boldsymbol{y}$ to a discretization of the transformed signal $G[f]$. Assuming that discretizing the signal commutes with the operator G, we denote this signal as $g(\boldsymbol{y}) \in Y^n$. While learning g does not suffer from the same identifiability issues as learning weight space transformations, it is tied to the resolution (set by the number of sample points n).

4 RESULTS

We illustrate the benefits of model editing via operator learning in two domains, computer vision and scientific machine learning, using different tasks.

In computer vision, our continuous signals are handwritten digits from the MNIST datasets (LeCun et al., 1998). Their neural representations are INRs (Sitzmann et al., 2020), and their representation as discretized signals corresponds to images in raster graphic formats. We focus on rotations as a transformation of interest. We compare instantiations of our framework based on Fourier neural operators (FNOs) and DeepONets with baselines that operate on weight space, such as a multilayer perceptron (MLP) and Deep Weight Space Network (DWSNet), as well as baselines that operate on discrete signals, such as a convolutional neural network (CNN) and a Vision Transformer (ViT). We compare the ability of these models to learn rotations by a fixed angle and their performance at subgrid resolution, finding that our approach achieves competitive performance on the former and outperforms baselines on the latter. We also show that our approach can be trained efficiently to generalize to arbitrary angles.

For scientific machine learning, we focus on the harmonic oscillator solutions as the continuous signal of interest and neural ODEs (Chen et al., 2018) as neural representations of the underlying vector field. We do not use discretized representations in our experiments. We study the task of mapping solutions of the undamped harmonic oscillator to solutions with nonzero damping. We instantiate our approach based on operator learning using FNOs and DeepONets and compare it to weight space baselines based on MLPs and DWSNets, showing that FNOs and DeepONets exhibit superior performance. A conceptual overview of models used is as follows:

- *FNO* combines learned transformations of Fourier coefficients with pointwise transformations of function values, allowing information to be exchanged across the input domain.
- *DeepONet* encodes the sampled input function through a branch network and output query coordinates through a trunk network, combining their features for predictions.
- *CNN* applies spatial convolutions to extract and combine image features, producing a transformed image.
- *ViT* represents an image as a sequence of patch embeddings and uses self-attention to exchange information between patches before reconstructing the output image.
- *DWSNet* processes network weights and biases as structured, layerwise objects using transformations equivariant to hidden-neuron permutations, producing edited parameters.
- The *MLP* processes flattened network parameters through fully connected layers to predict edited parameters, without explicitly encoding network structure or permutation symmetries.

4.1 ROTATING MNIST INRS

In this section, we describe our results on rotating MNIST INRs. The continuous signals f are thus handwritten digits, and the transformed signals $G[f]$ correspond to $180°$-rotated images. We create a dataset of SIREN INRs f_θ as neural representations of these continuous signals and their transformed versions (Sitzmann et al., 2020). We provide all details on INR training in Appendix A.1 and emphasize here that we trained INRs from a shared initialization to aid learning of the transformation in weight space.

Our operator learning approach is instantiated through the following **architectures**: an FNO and DeepONet. We use the following **baselines** for comparison: a CNN and ViT as conventional image-processing baselines as well as a DWSNet and MLP as weight-space baselines. We provide details

on training for both neural operators and baseline models in Appendix B.1 and report parameter counts in Tab. 8.

4.1.1 180° ROTATION

Task We evaluate all six models on the fixed 180° rotation task at the original 28×28 grid. All models are evaluated on the same held-out examples, using the exact 180° rotation of each source INR's reconstructed image. For the MLP and DWSNet, we render the INR defined by its predicted parameter vector before computing reconstruction error.

Results The ViT, FNO, and CNN achieve the lowest reconstruction error and most closely reproduce the target images (Figure 2, Table 4). The DeepONet and MLP also recover the rotated digits with relatively low error, although their predictions are noisier. In contrast, the DWSNet performs substantially worse, producing noisier reconstructions and a considerably broader distribution of errors than the rest of the models. These results establish that our approach based on operator learning, instantiated using FNO, performs competitively with strong computer vision baselines when evaluated at the resolution of the discretized signal these baselines are trained on. We find that, for this task, learning on weights performs less well than learning on discrete signals.

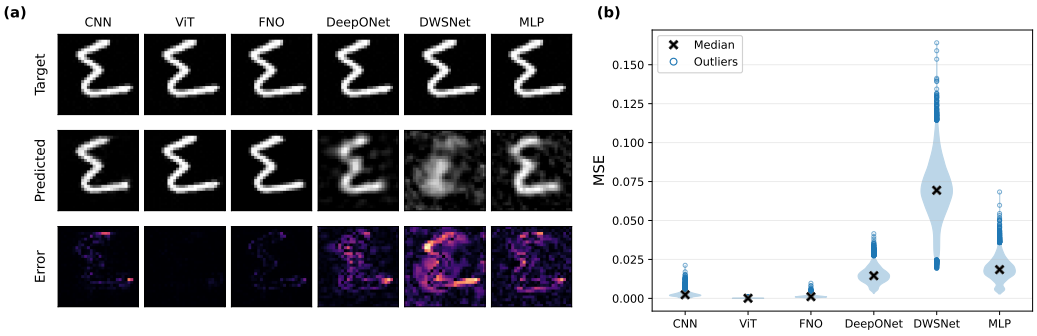

Figure 2: **(a):** Qualitative comparison on the 180° rotation task. Columns correspond to the CNN, ViT, DeepONet, FNO, DWSNet, and weight-space MLP. Rows show the target, model prediction, and absolute error. **(b):** Distribution of per-example MSE on the 180° rotation task. The ViT achieves the lowest overall error, followed by the FNO and CNN. Though higher-error, the DeepONet and MLP still perform well, while the DWSNet exhibits substantially higher error. This means our approach based on operator learning performs competitively with strong computer vision baselines applied to discretized signals.

4.1.2 GENERALIZATION TO UNSEEN COORDINATES

Task We next evaluate models trained to perform 180° rotations on regions between the coordinates used during training by sampling the represented functions at the midpoints of adjacent grid locations. This produces a 27×27 grid that was never observed during training.

Results Under this evaluation, the FNO exhibits the lowest mean squared error (MSE), with only a small increase in MSE relative to the original grid (Fig. 3, Tab. 5). Additionally, the MSE of DeepONet, DWSNet, and MLP barely increases, while CNN and ViT perform substantially worse, with ViT exhibiting the largest increase in MSE. These results highlight that our approach preserves the resolution independence of INR representations of continuous signals. While weight space approaches (DWSNet, MLP) also enjoy this benefit, they show overall higher MSE and more variable performance. As models trained on discretized signals (ViT, CNN) are tied to the original discretization, the observed drop in performance is expected.

4.1.3 GENERALIZATION ACROSS ANGLES

Architectures For this evaluation, we train an additional FNO and DeepONet model on signals that are rotated by randomly sampled angles. We emphasize that this training data is easy to collect,

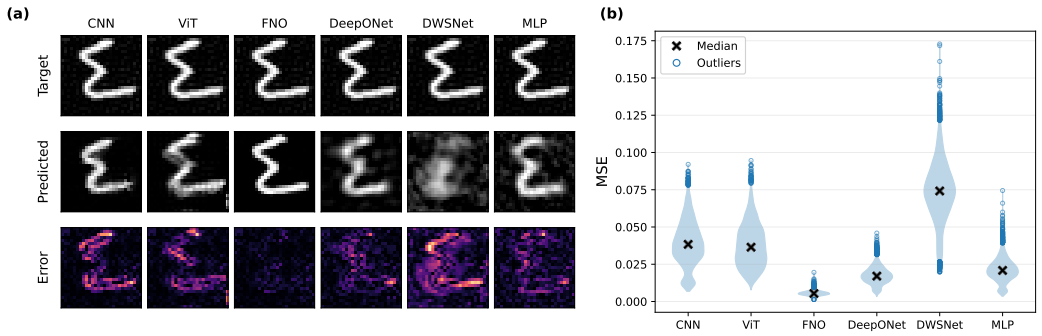

Figure 3: **(a):** Evaluation of the $180°$ rotation models at previously unseen spatial coordinates on a 27×27 grid formed by the midpoints between adjacent training-grid coordinates. **(b):** Distribution of per-sample MSE under sub-interval evaluation. The FNO, DeepONet, DWSNet, and MLP remain comparatively stable at unseen sampling locations, while the CNN and ViT exhibit a substantial increase in error. This shows that our approach based on operator learning preserves the resolution independence of neural representations of continuous signals.

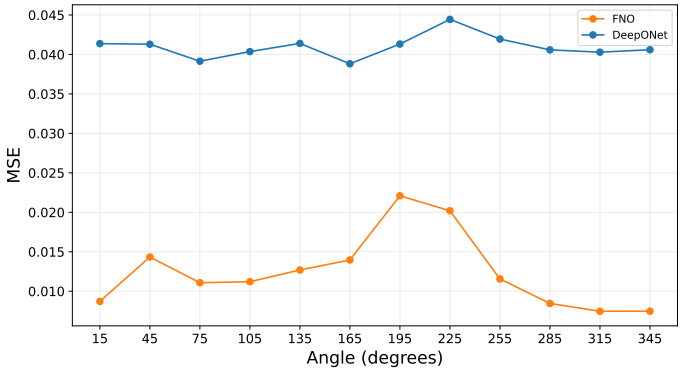

Figure 4: MSE of the FNO and DeepONet at rotation angles held out during training. The FNO shows lower MSE than DeepONet throughout. This illustrates the potential of our operator learning approach to model editing to generalize over parameterized families of transformations.

as the rotated signal $G[f]$ can be expressed in terms of the original signal f as $G[f](x) = f(Rx)$, where $R \in \mathbb{R}^{2\times2}$ is a rotation matrix. We do not include weight space baselines in this experiment, as creating the necessary training data would require training INRs for rotated digits at arbitrary angles, requiring excessive computational resources.

Task We evaluate the FNO and DeepONet, as instantiations of our operator learning approach to model editing, on rotations by angles excluded during training. Two sets of angles were withheld: a sparse set every $30°$, starting from $15°$, and a continuous arc from $195°$ to $225°$.

Result The FNO achieves substantially lower MSE than the DeepONet throughout the entire range of held-out angles (Fig. 4, Tab. 6). The FNO exhibits a pronounced increase in MSE from $195°$ to $225°$ where a continuous arc of angles was withheld from the training data, showing the importance of sufficient data density for generalization. A qualitative comparison (Fig. 5) shows that the FNO faithfully represents the rotated signal while the predictions of the DeepONet are blurry and follow the rotation only vaguely. This shows that the operator learning formulation of model editing (if instantiated using suitable architectures) can learn a parameterized family of transformations rather than only a fixed edit. Moreover, it illustrates that training is possible based on signals alone without the need to create weight space representations, which can be computationally intensive.

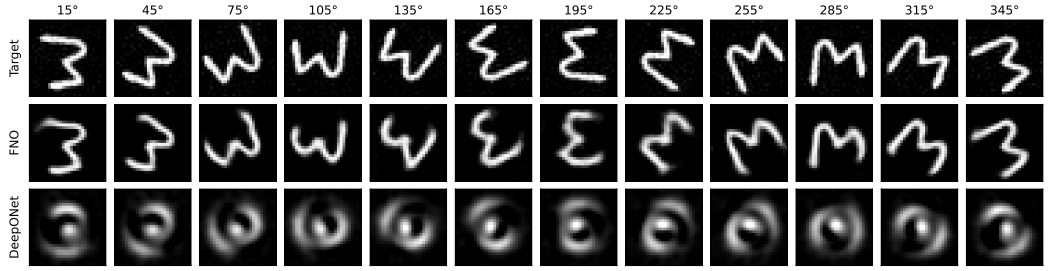

Figure 5: Qualitative evaluation of generalization to unseen rotation angles for a representative digit. Rows show the exact target, FNO prediction, and DeepONet prediction. The FNO predictions are faithful to the transformed signal and only degrade slightly over the continuous arc of held-out angles. In contrast, the DeepONet predictions are blurry. This shows that operator learning approaches to model editing can generalize over families of transformations when instantiated using suitable architectures.

4.2 Editing Neural ODEs

This section illustrates our approach in the context of scientific machine learning, in particular neural ODEs for the damped harmonic oscillator, i.e., the differential equation defined by the vector field

$$\Psi(t, x_1(t), x_2(t)) = \begin{pmatrix} x_2(t) \\ \varphi(t) - 2\zeta\omega x_2(t) - \omega^2 x_1(t) \end{pmatrix}, \tag{2}$$

where $x_1(t)$ is the position at time t, $x_2(t) = x_1'(t)$ is the velocity, i.e., time derivative of the position at time t, $\varphi : \mathbb{R} \to \mathbb{R}$ is a forcing function, $\omega > 0$ is the natural frequency, and $\zeta > 0$ specifies the damping. We train neural ODEs to encode this vector field, we describe our neural ODE training procedure in Appendix A.2.

We focus on learning the operator $G_{\zeta,\omega}[\cdot, \varphi]$ that takes a solution trajectory for the case without damping ($\zeta = 0$, simple harmonic oscillator (SHO)) to one with damping ($\zeta > 0$, damped harmonic oscillator (DHO)). We treat the system parameters ζ and ω, as well as the forcing function φ as known.

Our approach relies on neural operators that modify trajectories obtained by solving a neural ODE. Again, we use the following architectures: a FNO and DeepONet. We compare them against the following baselines that operate in weight space, and modify the weights of the neural ODE: a DWSNet and MLP. We provide further details on how neural operators and baselines are trained in Appendix B.2 and report parameter counts in Tab. 8.

Results Out of the four models, the FNO achieves smallest MSE, followed by DeepONet (Fig. 6, Tab. 7). The MLP and the DWSNet show substantially higher median MSE. However, their MSE distribution extends to cases lower than the median MSE of the FNO or DeepONet. These results showcase the strong performance of our neural operator-based model editing for neural ODEs. The differences between FNO and DeepONet may result from the Fourier series-based forcing functions used in our experiments that align well with the FNO's inductive biases. Moreover, we find that our neural operator-based approach can be trained more efficiently than weight space models that require, as we found in exploratory experiments, additional trajectory-level supervision, which makes additional numerical integration necessary.

5 Conclusion

In this work, we proposed viewing editing neural representations of continuous signals as an operator learning problem, and contrasted this approach with approaches that operate on model weights or discretized signals. We illustrated our approach with examples from computer vision and scientific computing.

Our neural operator-based approach enables editing INRs while preserving their resolution independence—a key desideratum that approaches based on discretized signals fail to meet. In

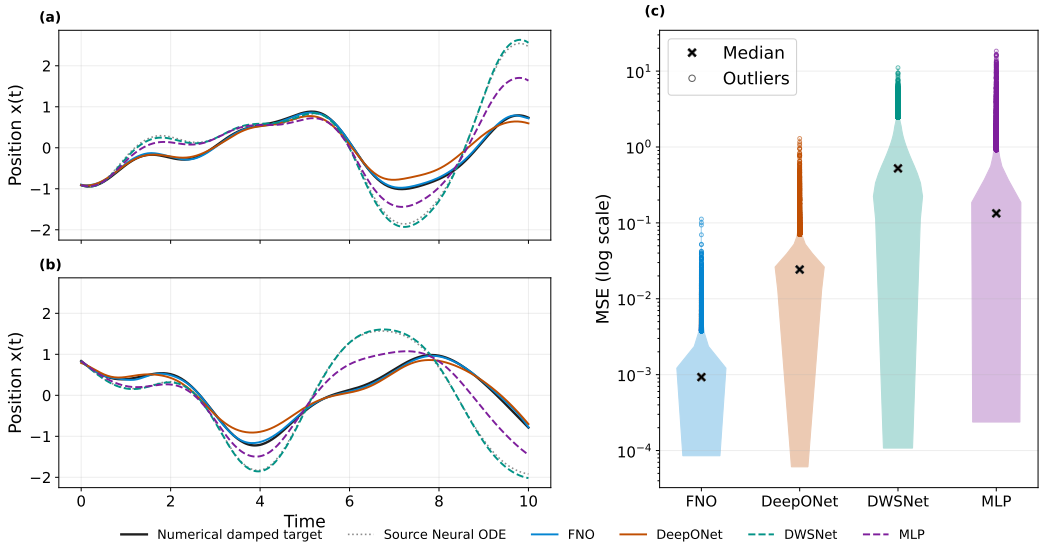

Figure 6: **(a),(b):** Evaluation of FNO (solid blue line), DeepONet (solid orange line), DWSNet (dashed green line), and MLP (dashed purple line) at previously unseen trajectories obtained as the solution to an unseen vector field. The FNO and DeepONet show best visual agreement. The MLP and DWSNet failed to learn the mapping. **(c):** Distribution of per-example MSE on the SHO to DHO mapping task. The distributions are plotted on log scale due to the high variance of performance.

addition, we showed that our approach, if instantiated with suitable neural operator architectures, can generalize across parametric families of transformations, e.g., across rotation angles. A key insight from this experiment is that our approach can be trained based on signals alone, whereas alternative approaches operating on weights require training additional neural representations, thus increasing computational burden substantially.

Finally, we showed that model editing as an operator learning problem is not restricted to INRs and geometric transformations. We were able to use neural operators to add damping terms to vector fields encoded in neural ODEs, achieving lower error and more consistent performance than baselines that operate in weight space.

Our experiments do, however, also have limitations as they are restricted to MNIST INR rotations and forced harmonic oscillators. Both settings use low-dimensional domains and well-defined transformations. Extending our approach to high-dimensional settings is an important direction for future work that requires defining suitable training protocols for neural operators in high-dimensional domains. In addition, testing our approach on real-world signals that exhibit less structured modifications is a crucial next step. Future work should also consider other neural representations.

Overall, viewing model editing as an operator learning task offers a promising alternative to methods that operate in weight space or on discretized signals, as the former suffer from non-identifiability while the latter break the resolution independence of neural representations. Our experiments point to neural operators as being effective, generalizable, and computationally efficient models for editing neural representations of continuous signals.

AI USE STATEMENT

In this work, we used generative AI tools to design and provide feedback on research methodology and experiments, and implement methods. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, generate synthetic datasets, to support qualitative and thematic data analysis, interpret results, formulate mathematical claims, propose or refine hypotheses or to clean and reformat datasets, or assist with translation. Providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs is not applicable to this work. Additionally, we used generative AI tools for suggesting experimental parameters, creating and edit-

ing software code, brainstorming, editing the research paper to improve readability, and identifying relevant literature. We have reviewed all AI-assisted work. We checked code and methods created with AI assistance for correctness, and verified that related work suggested by generative AI was accurate and relevant. For writing we only used AI for checking spelling and grammar, as well as to improve readability. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

We detail hyperparameters and experimental procedure in Appendix A, Appendix B and throughout Section 4. Our code is available at `https://anonymous.4open.science/r/operator-editing-F1BA`.

References

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging Models Modulo Permutation Symmetries. In *International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2209.04836.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, 2021.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf`.

Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-SDF: Conditional Generative Modeling of Signed Distance Functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2262–2272, 2023. URL `https://openaccess.thecvf.com/content/ICCV2023/html/Chou_Diffusion-SDF_Conditional_Generative_Modeling_of_Signed_Distance_Functions_ICCV_2023_paper.html`.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92(3):88, July 2022. ISSN 1573-7691. doi: 10.1007/s10915-022-01939-z. URL `https://doi.org/10.1007/s10915-022-01939-z`.

Pan Du, Meet Hemant Parikh, Xiantao Fan, Xin-Yang Liu, and Jian-Xun Wang. Conditional neural field latent diffusion model for generating spatiotemporal turbulence. *Nature Communications*, 15 (1):10416, November 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-54712-1. URL `https://www.nature.com/articles/s41467-024-54712-1`.

Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 5694–5725. PMLR, June 2022a. URL `https://proceedings.mlr.press/v162/dupont22a.html`.

Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. Generative Models as Distributions of Functions. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, pp. 2989–3015. PMLR, May 2022b. URL `https://proceedings.mlr.press/v151/dupont22a.html`.

Sidaty El Hadramy, Nazim Haouchine, Michael Wehrli, and Philippe C. Cattin. NOIR: Neural Operator mapping for Implicit Representations, March 2026. URL `http://arxiv.org/abs/2603.13118`.

Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. HyperDiffusion: Generating Implicit Neural Fields with Weight-Space Diffusion. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14254–14264, October 2023. doi:

10.1109/ICCV51070.2023.01315. URL https://ieeexplore.ieee.org/abstract/document/10378032.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit Behavioral Cloning. In *Proceedings of the 5th Conference on Robot Learning*, pp. 158–168. PMLR, January 2022. URL https://proceedings.mlr.press/v164/florence22a.html.

Saumya Gupta, Scott Biggs, Moritz Laber, Zohair Shafi, Robin Walters, and Ayan Paul. DeepWeightFlow: Re-Basined Flow Matching for Generating Neural Network Weights. In *International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2601.05052.

Jan Hagnberger, Marimuthu Kalimuthu, and Mathias Niepert. Vectorized Conditional Neural Fields: A Framework for Solving Time-dependent PDEs. In *ICLR 2024 Workshop on AI4DifferentialEquations In Science*, March 2024. URL https://openreview.net/forum?id=qMLjDCKF8N.

Xiaolong Han, Zehong Wang, Bo Zhao, Binchi Zhang, Jundong Li, Damian Borth, Rose Yu, Haggai Maron, Yanfang Ye, Lu Yin, and Ferrante Neri. A Survey of Weight Space Learning: Understanding, Representation, and Generation, 2026.

Vincent Herrmann, Francesco Faccio, and Jürgen Schmidhuber. Learning Useful Representations of Recurrent Neural Network Weight Matrices. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 18205–18227. PMLR, 2024.

Wonbong Jang and Lourdes Agapito. CodeNeRF: Disentangled Neural Radiance Fields for Object Categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12949–12958, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Jang_CodeNeRF_Disentangled_Neural_Radiance_Fields_for_Object_Categories_ICCV_2021_paper.html.

Jonathan Kahana, Eliahu Horwitz, Imri Shuval, and Yedid Hoshen. Deep Linear Probe Generators for Weight Space Learning. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.10811.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, February 2022. URL https://arxiv.org/pdf/2202.02435.pdf.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural Operator: Learning Maps Between Function Spaces with Applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://doi.org/10.1109/5.726791.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2021. doi: 10.48550/arXiv.2010.08895.

Derek Lim, Haggai Maron, Marc T. Law, Jonathan Lorraine, and James Lucas. Graph Metanetworks for Processing Diverse Neural Architectures. In *International Conference on Learning Representations*, 2024.

Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing Conditional Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5753–5763, October 2021. doi: 10.1109/ICCV48922.2021.00572. URL https://ieeexplore.ieee.org/abstract/document/9710143.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5.

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. *Communications of the ACM*, 65(1):99–106, December 2021. ISSN 0001-0782. doi: 10.1145/3503250. URL https://dl.acm.org/doi/10.1145/3503250.

Aviv Navon, Aviv Shamsian, Idan Achituve, Ethan Fetaya, Gal Chechik, and Haggai Maron. Equivariant Architectures for Learning in Deep Weight Spaces. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25790–25816. PMLR, 2023.

Sourav Pal, Harshavardhan Adepu, Clinton Wang, Polina Golland, and Vikas Singh. Implicit Representations via Operator Learning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 39022–39041. PMLR, July 2024. URL https://proceedings.mlr.press/v235/pal24a.html.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174, June 2019. doi: 10.1109/CVPR.2019.00025. URL https://ieeexplore.ieee.org/document/8954065.

M. Raissi, P. Perdikaris, and G. E. 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, February 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Aviv Shamsian, Aviv Navon, David W. Zhang, Yan Zhang, Ethan Fetaya, Gal Chechik, and Haggai Maron. Improved Generalization of Weight Space Networks via Augmentations. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 44378–44393. PMLR, 2024.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473, 2020.

Jakub M. Tomczak. *Deep Generative Modeling*. Springer International Publishing, Cham, 2022. ISBN 978-3-030-93157-5 978-3-030-93158-2. doi: 10.1007/978-3-030-93158-2. URL https://link.springer.com/10.1007/978-3-030-93158-2.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum*, 41(2):641–676, 2022. doi: 10.1111/cgf.14505.

Allan Zhou, Kaien Yang, Yiding Jiang, Kaylee Burns, Winnie Xu, Samuel Sokota, J. Zico Kolter, and Chelsea Finn. Neural Functional Transformers. In *Advances in Neural Information Processing Systems*, volume 36, pp. 77485–77502, 2023. doi: 10.52202/075280-3387.

Allan Zhou, Chelsea Finn, and James Harrison. Universal Neural Functionals. In *Advances in Neural Information Processing Systems*, volume 37, pp. 104754–104775, 2024. doi: 10.52202/079017-3326.

A Model Zoos

A.1 INRs

We construct the INR dataset from $60,000$ images in the original MNIST training split, without including the official MNIST test split (LeCun et al., 1998). Pixel intensities are normalized to $[0, 1]$, and pixel coordinates are mapped to a uniform 28×28 grid over $[-1, 1]^2$. For each image, we independently fit two INRs: one to the original and one to the 180° rotation. The resulting dataset contains $60,000$ pairs, or $120,000$ fitted INRs.

Each INR is a SIREN (Sitzmann et al., 2020) with architecture $2 \to 32 \to 32 \to 1$, sinusoidal hidden activations with frequency factor $\omega_0 = 30$, and a linear output layer, giving $1,185$ trainable parameters. All source and rotated INRs start from the same global initialization, generated with seed 12345. This controls initialization differences across the parameter bank, although it does not eliminate the non-uniqueness of neural parameterizations.

We fit the INRs using pixelwise mean squared error and Adam with learning rate 5×10^{-4}. Fitting is performed in batches of 256 pairs, corresponding to 512 independent INRs, using CUDA mixed precision. Each INR is optimized for at most $1,000$ steps and is stopped when its reconstruction reaches 40 dB peak signal-to-noise ratio, checked every ten steps. The dataset ordering uses seed 7. We retain the fitted weights and biases, source-image identifiers, digit labels, and reconstruction metrics.

For the editor experiments, the bank is shuffled using seed 42. The first $40,000$ pairs are used for training and the next $5,000$ for validation. The fixed-angle training notebooks reserve the remaining $15,000$ pairs for testing; the comparison figures evaluate the first $10,000$ of these pairs. The arbitrary-angle experiments explicitly reserve $10,000$ test examples and evaluate the first $5,000$ at each queried angle.

A.2 Neural ODEs

We generate $1,500$ forced oscillator systems. For each system, we sample a natural frequency $\omega \sim \mathcal{U}(0.5, 3.0)$ and a damping ratio $\zeta \sim \mathcal{U}(0.05, 0.4)$, where $\mathcal{U}(a, b)$ denotes the uniform distribution on the interval (a, b). Each system has 40 trajectories with independently sampled initial position and velocity from $\mathcal{U}(-1, 1)$.

Forcing functions are sampled as five-mode Fourier series. Amplitudes follow a standard normal distribution, phases are uniform on $[0, 2\pi]$, and angular frequencies are sampled between 0.25ω and 3ω. Each forcing function is normalized by its root mean square (RMS) magnitude and is subsequently set to zero with probability 0.1. This means approximately 10% of examples are unforced simple harmonic oscillators. We generate paired undamped and damped trajectories using the `DOP853` Runge-Kutta method as implemented in `scipy` (Virtanen et al., 2020), with forcing linearly interpolated between samples. Trajectories are recorded at 256 uniformly spaced times over $[0, 10]$.

For each physical system, we fit separate source and target neural ODEs to the undamped and damped vector fields. Each vector field is represented as an MLP with architecture $3 \to 32 \to 32 \to 2$, two $\tanh$ hidden layers, and a linear output layer. Its inputs are position, velocity, and forcing, and its outputs are the two state derivatives. Each network contains $1,250$ trainable parameters. The source and target networks within a pair share their initial parameters, while different pairs are initialized independently.

Vector field fitting minimizes MSE against the state derivatives evaluated at sampled trajectory states. We use Adam with learning rate 3×10^{-3} for 400 steps, processing 32 systems at a time and sampling 128 points per model at each step. Source trajectories are obtained by integrating the fitted source networks using a PyTorch implementation of the fourth-order Runge-Kutta (RK4) method, with forcing values at intermediate stages computed by linear interpolation between adjacent time samples.

Dataset generation and splitting used seed 0. We split by physical system into $1,000$ training, 125 validation, and 375 test systems. All 40 trajectories from a system remain in the same partition, giving $40,000$ training, $5,000$ validation, and $15,000$ test trajectories.

B EXPERIMENTAL DETAILS

B.1 MNIST ROTATION EXPERIMENTS

All MNIST editors are trained for 20 epochs with learning rate 10^{-3} and seed 42, without a learning-rate scheduler. We select the checkpoint with the lowest validation image MSE. Table 1 summarizes the architecture and optimizer settings.

Table 1: Architecture and training settings for the MNIST editors. FNO and DeepONet use separate checkpoints for fixed-angle and arbitrary-angle rotation.

Model	Architecture	Activation	Optimizer	Batch size	Weight decay
CNN	Six width-128 convolutional blocks	GELU	AdamW	256	10^{-5}
ViT	Six width-128 encoder layers; four heads; FFN width 256	GELU	AdamW	256	10^{-5}
FNO	Six 2D Fourier layers; width 128; modes $(12, 12)$	GELU	Adam	256	0
DeepONet	Three linear layers per branch/trunk; latent width 128	GELU	Adam	256	0
DWSNet	Width 64; 3 hidden layers; mean aggregation	ReLU	AdamW	64	10^{-5}
MLP	Three hidden layers of width 2,048	GELU	AdamW	256	10^{-5}

The CNN uses 1×1 lifting and output convolutions, with each intermediate block summing a dilated 3×3 spatial convolution and a 1×1 pointwise convolution. The dilation factors are $(1, 2, 4, 8, 4, 2)$. The ViT uses non-overlapping 2×2 patches. Both CNN and ViT receive image intensities augmented with normalized x- and y-coordinate channels.

The FNO input consists of source intensities and spatially broadcast $\cos\theta$ and $\sin\theta$ channels. The FNO projects its final features to a scalar intensity at each grid point. DeepONet's branch encodes the flattened source samples and $(\cos\theta, \sin\theta)$, while its trunk encodes query coordinates using the raw coordinates and sine/cosine features at frequencies $\pi, 2\pi, 4\pi, 8\pi$. Their outputs are combined by an inner product and a learned bias.

DWSNet applies permutation-equivariant layers to the structured weight matrices and bias vectors. The MLP maps a flattened 1185-dimensional source parameter vector to an output vector of the same size. Both weight-space editors operate on parameters standardized coordinate-wise using training-set statistics, and their outputs are transformed back the original parameter scales before rendering.

For the intermediate-coordinate experiment, we evaluate the same fixed-angle checkpoints on the 27×27 grid formed by midpoints between adjacent training coordinates. DeepONet retains its original 28×28 branch sensors and evaluates its trunk at the new coordinates.

For arbitrary rotation, one angle is sampled per training batch. Targets are generated by evaluating the source INR at inverse-rotated coordinates, with zero intensity outside $[-1, 1]^2$. The angle partitions are given in Tab. 2.

Table 2: Rotation-angle partitions for the arbitrary-angle experiment.

Partition	Angles
Training	319 integer angles in $[0^\circ, 359^\circ]$, excluding $\{15^\circ, 45^\circ, \ldots, 345^\circ\}$ and angles strictly inside $(195^\circ, 225^\circ)$.
Validation	$\{0^\circ, 30^\circ, 60^\circ, 90^\circ, 120^\circ, 150^\circ, 180^\circ, 240^\circ, 270^\circ, 300^\circ, 330^\circ\}$.
Sparse held-out	$\{15^\circ, 45^\circ, \ldots, 345^\circ\}$: 12 angles.
Continuous arc	$\{195.5^\circ, 196.5^\circ, \ldots, 224.5^\circ\}$: 30 angles.

The endpoints 195° and 225° are also excluded from training because they belong to the sparse held-out set.

B.2 NEURAL ODE EXPERIMENTS

All four editors use the same neural-ODE bank and system-level partitions. They are trained with Adam for 20 epochs at learning rate 10^{-3}, with no weight decay or learning-rate scheduler. All computations use float32. We select checkpoints by the lowest validation MSE against the numerical damped trajectories. Further details are reported in Tab. 3.

Table 3: Architecture and training settings for the neural-ODE editors. Batch sizes count trajectories for function-space models and neural-ODE pairs for weight-space models. All models use a learning rate of 10^{-3} and train for 20 epochs.

Model	Architecture	Activation	Optimizer	Batch size
FNO	Four 1D Fourier layers; width 32; modes setting 16	GELU	Adam	64
DeepONet	Branch: $770 \to 128 \to 128 \to 256$ Trunk: $1 \to 128 \to 128 \to 128$	GELU	Adam	64
DWSNet	Width 32; two hidden DWS layers; mean aggregation	ReLU	Adam	32
MLP	$1251 \to 512 \to 512 \to 1250$	GELU	Adam	64

DWSNet has three DWS layers in total, including its output layer.

The FNO receives six temporal channels: source position, source velocity, forcing, time, natural frequency, and damping ratio. DeepONet's branch receives the source trajectory, forcing history, natural frequency, and damping ratio, while its trunk receives the time coordinate. Both models directly predict the damped position and velocity.

The weight-space editors receive the source vector-field parameters and the requested damping ratio. They predict additive changes $\Delta\theta$, giving $\theta_{\text{edited}} = \theta_{\text{source}} + \Delta\theta$. Predictions are obtained by integrating the resulting neural ODE on the full temporal grid using RK4. For weight-space training, the loss weighs both the parameters and actual numerical prediction, that is:

$$\mathcal{L} = 0.1\mathcal{L}_{\text{param}} + \mathcal{L}_{\text{trajectory}} \tag{3}$$

C Additional Results

We report the mean, median, and population standard deviation of the per-example errors used in comparison plots. For our experiments on MNIST INRs (Tab. 4-Tab. 6), an observation is one test image at a specified grid and rotation angle. We report mean squared error (MSE) and mean absolute error (MAE). For the neural ODE experiment (Tab. 7), an observation is one held-out system/trajectory pair, with MSE and MAE averaged over all time points and both state components. We report parameter counts in Tab. 8.

Table 4: Fixed 180° MNIST rotation on the original 28×28 grid. Statistics summarize 10,000 test images per model.

	MSE			MAE		
Model	**Mean**	**Median**	**Std.**	**Mean**	**Median**	**Std.**
CNN	0.002678	0.002325	0.001693	0.022452	0.021903	0.006645
ViT	0.000080	0.000067	0.000048	0.006083	0.005908	0.000864
FNO	0.001230	0.001104	0.000661	0.014932	0.014623	0.003296
DeepONet	0.014824	0.014543	0.005020	0.076699	0.077173	0.014890
DWSNet	0.069275	0.069399	0.018720	0.196534	0.198965	0.031372
MLP	0.018824	0.018464	0.007164	0.095064	0.096174	0.018492

Table 5: Fixed 180° MNIST rotation on the intermediate 27×27 grid. Statistics summarize the same 10,000 test images used for the original-grid evaluation.

	MSE			MAE		
Model	**Mean**	**Median**	**Std.**	**Mean**	**Median**	**Std.**
CNN	0.038926	0.038346	0.015106	0.110088	0.110825	0.027153
ViT	0.037786	0.036442	0.014274	0.113553	0.114024	0.024430
FNO	0.005441	0.005387	0.001641	0.054359	0.054881	0.008924
DeepONet	0.017290	0.017037	0.005631	0.092845	0.093837	0.016211
DWSNet	0.073825	0.074172	0.019929	0.203923	0.206753	0.032434
MLP	0.021167	0.020866	0.007755	0.105769	0.107131	0.019598

Table 6: Arbitrary-angle MNIST rotation. Statistics pool 5,000 test images across 12 sparse held-out angles or 30 continuous-arc angles, giving 60,000 and 150,000 image–angle observations per model, respectively.

		MSE			MAE		
Angle set	**Model**	**Mean**	**Median**	**Std.**	**Mean**	**Median**	**Std.**
Sparse held-out	FNO	0.012434	0.010693	0.007215	0.057455	0.055924	0.015165
Sparse held-out	DeepONet	0.040963	0.039870	0.010765	0.122426	0.122117	0.019829
Continuous arc	FNO	0.026653	0.024859	0.010792	0.080528	0.078941	0.018854
Continuous arc	DeepONet	0.043547	0.042255	0.011617	0.125512	0.124860	0.021282

Table 7: Neural-ODE damped-trajectory editing. Statistics summarize 15,000 trajectories from 375 held-out systems. Errors are averaged over time and both state components within each trajectory.

	MSE			MAE		
Model	**Mean**	**Median**	**Std.**	**Mean**	**Median**	**Std.**
FNO	0.001879	0.000929	0.003404	0.028883	0.023687	0.017322
DeepONet	0.036868	0.024272	0.055676	0.129229	0.116828	0.068018
DWSNet	0.836236	0.522581	0.934621	0.613673	0.560727	0.332182
MLP	0.451458	0.133530	1.040159	0.387636	0.285957	0.330933

Table 8: Trainable parameter counts, measured in real scalar components. Each complex-valued parameter contributes two real components. Dashes indicate models not used in the corresponding experiment.

	Parameters	
Model	**MNIST**	**Neural ODE**
CNN	985,217	—
ViT	797,316	—
FNO	16,782,081	87,202
DeepONet	169,217	181,506
DWSNet	1,142,271	301,139
MLP	13,249,697	1,544,930