CONTRASTIVE DIFFUSION ALIGNMENT: LEARNING STRUCTURED LATENTS FOR CONTROLLABLE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models excel at generation, but their latent spaces are high dimensional and not explicitly organized for interpretation or control. We introduce ConDA (Contrastive Diffusion Alignment), a plug-and-play geometry layer that applies contrastive learning to pretrained diffusion latents using auxiliary variables (e.g., time, stimulation parameters, facial action units). ConDA learns a low-dimensional embedding whose directions align with underlying dynamical factors, consistent with recent contrastive learning results on structured and disentangled representations. In this embedding, simple nonlinear trajectories support smooth interpolation, extrapolation, and counterfactual editing while rendering remains in the original diffusion space. ConDA separates editing and rendering by lifting embedding trajectories back to diffusion latents with a neighborhood-preserving k-NN decoder and is robust across inversion solvers. Across fluid dynamics, neural calcium imaging, therapeutic neurostimulation, facial expression dynamics, and monkey motor cortex activity, ConDA yields more interpretable and controllable latent structure than linear traversals and conditioning-based baselines, indicating that diffusion latents encode dynamics-relevant structure that can be exploited by an explicit contrastive geometry layer.

1 INTRODUCTION

Controlling generation of complex system dynamics, such as how fluid flows evolve around an obstacle, how neural activity responds to therapeutic stimulation, or how facial expressions change, requires generative models whose latent spaces have a geometry that captures trajectories and allows movement along time and other control variables in a principled way. Despite diffusion models' success in generative modeling, a fundamental limitation remains: their latent spaces are not organized for such complex processes, leaving no principled way to traverse temporal or condition-dependent changes. Addressing this gap is crucial for applications where control is as important as fidelity.

Diffusion models provide unmatched fidelity, stable training, and reliable inversion (Ho et al., 2020; Dhariwal and Nichol, 2021; Rombach et al., 2022; Mokady et al., 2023). Yet their latents are not dynamics-aware: linear interpolations often produce implausible intermediate states (Wang and Golland, 2023; Hahm et al., 2024), and conditioning mechanisms such as ControlNet or InstructPix2Pix (Zhang et al., 2023; Brooks et al., 2023) guide samples but do not yield consistent traversal directions across time or conditions. As we show in Section 5, these limitations result in blurry or entangled transitions when interpolating fluid dynamics trajectories and inconsistent progression in neural recordings and facial expressions (Figs. 2).

Recent advances attempt to address this challenge. Geometry-preserving traversals (Wang and Golland, 2023; Hahm et al., 2024), physics-informed priors (Shu et al., 2023), and video diffusion models such as MCVD (Voleti et al., 2022), Imagen Video (Ho et al., 2022a), and Make-A-Video (Singer et al., 2022) improve interpolation or temporal realism, but they do not organize diffusion latents for controllable dynamics. The missing piece is a general recipe for dynamics-aware diffusion: a way to align latent geometry with system variables while preserving generative fidelity. Our experiments confirm that geometry-preserving and conditioning-based methods still fail to yield smooth or interpretable traversal of dynamics across domains.

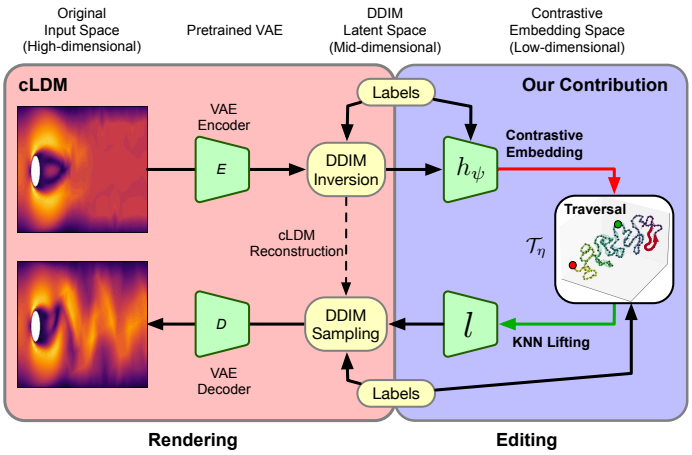


Figure 1: Method overview of ConDA framework. We encode a high-dimensional sequence of frames into a diffusion model feature latent trajectory via a diffusion model (e.g. cLDM) and diffusion inversion (e.g., DDIM/Rex-RK4), then map these latents into a compact, contrastively learned embedding space that organizes local geometry to reflect nonlinear dynamics. An editing operator modifies the trajectory in the embedding space under new conditions (local traversal, interpolation, or class transfer). The edited embedding trajectory is lifted back to the feature latent space (via a learned decoder or kNN neighborhood interpolation) and decoded frame-by-frame by the cLDM's conditional generator, yielding a new output sequence with controlled dynamics and preserved diversity and fidelity.

We introduce ConDA (Contrastive Diffusion Alignment), a flexible framework that addresses this limitation. ConDA organizes pretrained diffusion latents into a compact, contrastively-learned embedding space using auxiliary variables such as time, labels, or stimulation parameters. Within this space, now standard nonlinear operators such as splines, finite differences, and LSTMs enable smooth trajectory traversal, while the original diffusion latents are used for rendering to preserve fidelity. This separation of *editing* and *rendering* (see Figure 1 and Section 3) turns diffusion models into controllable generators of nonlinear dynamics.

Across five different domains, ConDA consistently improves controllability while preserving fidelity. In fluid dynamics, ConDA achieves higher reconstruction quality (35.7 PSNR vs. 28.3 for linear baselines) and smoother flow-field interpolations (Fig. 2). In neural calcium imaging, it produces smoother temporal progression across activity states (Fig. 2). In neurostimulation, it captures class-dependent transitions that vary systematically with a given stimulation coil angle (Fig. 3). In facial expression dynamics, it preserves subject identity while enabling smooth traversal of expressions (Fig. 4). In monkey motor control, it organizes condition-dependent dynamics in delayed reaching tasks (Fig. 5). Together, these results demonstrate that contrastive organization transforms simple nonlinear operators into effective tools for controlling diffusion latents across spatiotemporal systems.

Our main contributions are: 1. We propose ConDA, a framework for dynamics-aware diffusion that separates *editing* in a compact, contrastively-learned embedding space from *rendering* in the original diffusion latents. 2. We show that standard nonlinear traversal operators such as splines, finite differences, and LSTMs become effective when applied in this structured space, enabling smooth trajectory traversal beyond linear interpolation or conditioning. 3. We validate ConDA across five spatiotemporal domains (fluid dynamics, neurostimulation, neural calcium imaging, facial expression dynamics, and monkey motor control) demonstrating improved temporal consistency, controllable condition-dependent transitions, and identity preservation.

2 Related Work

Diffusion. Diffusion models are now the leading framework for high-fidelity image and video generation (Ho et al., 2020; Song et al., 2021; Dhariwal and Nichol, 2021; Rombach et al., 2022), offering photorealistic outputs, stable training, and reliable inversion (Mokady et al., 2023). However, their latent spaces are not explicitly structured for dynamics: linear interpolations often yield unreal-

istic transitions (Wang and Golland, 2023; Hahm et al., 2024), and conditioning methods such as ControlNet or InstructPix2Pix guide edits without ensuring consistent traversal directions (Zhang et al., 2023; Brooks et al., 2023).

Controllability. Efforts to improve controllability include geometry-preserving traversals (Wang and Golland, 2023; Hahm et al., 2024), physics-informed priors (Shu et al., 2023; Shan et al., 2024), and video diffusion models (Ho et al., 2022b; Voleti et al., 2022; Singer et al., 2022) that emphasize realism over latent geometry. Time-series approaches such as Diffusion-TS (Yuan and Qiao, 2024) and diffusion for sequential data (Liu et al., 2021), as well as label-efficient segmentation using diffusion (Baranchuk et al., 2022), remain task-specific rather than general-purpose.

GAN editing. GANs revealed interpretable latent directions (Radford et al., 2015; Karras et al., 2020; Shen et al., 2020), inspiring disentanglement strategies such as InfoGAN and β-VAE (Chen et al., 2016; Higgins et al., 2017). Yet GAN inversion is imperfect (Abdal et al., 2019; Xia et al., 2022), latent directions are often linear, and diffusion editing typically remains prompt- or attention-based (Meng et al., 2021; Avrahami et al., 2022; Hertz et al., 2022) rather than organizing latent space into interpretable trajectories.

Dynamics models. Sequence models such as the Kalman filter (Kalman, 1960), SINDy (Brunton et al., 2016), structured SSMs (HiPPO, S4) (Gu et al., 2020; 2021; 2022), and neural ODE/SDE approaches (Rubanova et al., 2019; Kidger et al., 2021b) provide principled dynamics. Diffusion has also been adapted for forecasting and imputation (Rasul et al., 2021; Liu et al., 2021; Shi et al., 2021), but applying these directly to diffusion latents often yields unstable or uninterpretable trajectories. ConDA instead provides a geometry layer that aligns latents with nonlinear traversal.

Disentanglement and contrastive learning. Contrastive objectives structure embeddings (Oord et al., 2018; Chen et al., 2020; Poole et al., 2018; Wang and Isola, 2020), with applications to neuroscience (Schneider et al., 2023) and identifiability guarantees (Lyu and Fu, 2022; Cui et al., 2022). Diffusion disentanglement (Shen et al., 2022; Hahm et al., 2024) largely targets static factors. ConDA extends this work by using auxiliary variables (e.g., time, stimulation, expression) to align latents with spatiotemporal processes, enabling smooth nonlinear trajectory traversal.

Summary. Prior work advances interpolation, video realism, and disentanglement, but none provide an identifiable framework for organizing pretrained diffusion latents for nonlinear, interpretable, controllable generation. ConDA fills this gap by (1) using contrastive learning with auxiliary variables to structure embeddings, (2) applying nonlinear traversal operators in this space, and (3) separating compact embedding edits from rendering in diffusion latents. Our experiments show improved temporal consistency, smooth trajectories, and faithful reconstructions across spatiotemporal data.

3 METHOD

3.1 NOTATION AND PROBLEM STATEMENT

We consider controlled sequence generation for high-dimensional spatiotemporal data (e.g., videos of fluid dynamics, calcium imaging videos, neurostimulation E-fields, monkey electrophysiology, facial expression sequences). Each sequence $x_{1:S} = (x_1, \ldots, x_S)$ consists of a series of frames (e.g., images) $x_s \in \mathcal{X}$ with associated per-frame auxiliary variables $y_s \in \mathcal{Y}$ (e.g., encoding time, coil angle, facial action units, etc.). Our goal is to generate a *target sequence* $\hat{x}_{j+1:j+n} = (\hat{x}_{j+1}, \ldots, \hat{x}_{j+n})$ frame-by-frame under a user-specified auxiliary-variable trajectory $y'_{j+1:j+n}$ (e.g., a single target image is the special case $n = 1$). We study three cases: (i) *reconstruction*, where $y'_{j+1:j+n} = y_{j+1:j+n}$ and the target indices coincide with observed indices (we regenerate observed frames under observed conditions to assess "round-trip" reconstruction quality); (ii) *interpolation and counterfactual editing*, where we define a contiguous held-out block of indices $J = \{j+1, \ldots, j+n\} \subset \{1, \ldots, S\}$, exclude $\{x_s : s \in J\}$ from the training set, and at test time generate $\hat{x}_J$ under an interpolated or counterfactual auxiliary-variable trajectory y'_J given the remaining frames $\{x_s : s \notin J\}$; and (iii) n-step-ahead *prediction*, where given training data $(x_{1:s}, y_{1:s})$ we set $J = \{s+1, \ldots, s+n\}$, hold these indices out, and generate $\hat{x}_J$ under an extrapolated auxiliary time trajectory y_J beyond the training data. In all three cases during training, we train on contiguous batches of individual frame–auxiliary-variable pairs (x_s, y_s) drawn from sequences.

ConDA proceeds in three stages: (1) train a diffusion model (e.g., cLDM) and use diffusion inversion (e.g., DDIM, Rex-RK4) to map each observation–auxiliary-variable pair (x_s, y_s) to a feature latent z_s; (2) learn a compact contrastive embedding $c_s = h_\psi(z_s, y_s)$ of the feature latent space whose local geometry aligns with the auxiliary variables; and (3) at test time, traverse the embedding trajectory $(c_1, \ldots, c_S)$ to either (a) reconstruct observed data, (b) interpolate/edit held out data points, or (c) predict forward in time, and then lift points edited for (a), (b), or (c) back to the feature latent space with a kNN decoder, and render the corresponding sequence with the diffusion model decoder.

Importantly, we define two latent spaces that serve different purposes in our generation pipeline: **(1) Feature Latent Space ($\mathcal{Z}$):** This relatively high-dimensional space (diffusion model latent space) is designed for near-lossless image synthesis. We train a conditional decoder $f_\theta : \mathcal{Z} \times \mathcal{Y} \to \mathcal{X}$ to synthesize frames $\hat{x}_s = f_\theta(z_s, y_s)$. An inverse mapping $g_\phi : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ is also learned to project observed frames back into this space, such that $z_s = g_\phi(x_s, y_s)$ and $\hat{x}_s \approx f_\theta(z_s, y_s) \approx x_s$. While $\mathcal{Z}$ supports high–fidelity frame synthesis, it remains high–dimensional (e.g., $\dim(\mathcal{Z}) = 32 \times 32 \times 4$ when using cLDM on an image frame with $\dim(\mathcal{X}) = 256 \times 256 \times 3$) and its geometry is not explicitly organized for smooth temporal/conditioned traversal; direct editing in $\mathcal{Z}$ is compute–intensive and brittle. **(2) Compact Structured Embedding ($\mathcal{C}$):** This is a low-dimensional embedding of $\mathcal{Z}$, with $\dim(\mathcal{C}) \ll \dim(\mathcal{Z})$, explicitly designed for stable, interpretable trajectory manipulation. We introduce a mapping $h_\psi : \mathcal{Z} \times \mathcal{Y} \to \mathcal{C}$ such that $c_s = h_\psi(z_s, y_s)$. The mapping h_ψ is intentionally lossy, preserving only the local neighborhood geometry and the essential factors governing the intrinsic dynamics and conditioned state. As we will show, this structured, low-dimensional representation is good for dynamic modeling and condition-controlled modifications.

The Generation Pipeline. The overall synthesis pipeline proceeds in several steps. First, *Encoding:* An observed sequence $(x_{1:S}, y_{1:S})$ is encoded into the feature latent space as $z_{1:S} = g_\phi(x_{1:S}, y_{1:S})$. Second, *Structured Embedding:* The feature latents are then mapped to the compact structured space, $c_{1:S} = h_\psi(z_{1:S}, y_{1:S})$, where efficient editing operations can be performed. Third, *Editing:* An editing operator $\mathcal{T}_\eta : \mathcal{C} \times \mathcal{Y} \times \mathcal{Y} \to \mathcal{C}$ is applied to modify the dynamics trajectory based on new conditions. For each step s, the embedding is updated as $\hat{c}_{s'} = \mathcal{T}_\eta(c_s; y_s \to y_{s+\Delta s})$ for a local traversal along the trajectory and $\hat{c}'_s = \mathcal{T}_\eta(c_s; y_s \to y'_s)$ for a jump to a new target condition (e.g., class transfer). Without loss of generality, we use a single notation below, $\hat{c}_{s'}$, to denote the edited embedding, covering both the local traversal and the jump to a new target condition. Fourth, *Lifting:* The edited dynamics embedding $\hat{c}_{s'}$ is lifted back to the high-dimensional feature space, yielding $\hat{z}_{s'}$ by a learned decoder $\ell : \mathcal{C} \to \mathcal{Z}$ or via a k-nearest neighbor (kNN) decoder:

$$\ell(\hat{c}_{s'}) = \sum_{j \in \mathcal{N}_K(\hat{c}_{s'})} w_j(\hat{c}_{s'})\, z_j, \quad \text{where } w_j(\hat{c}_{s'}) \propto \frac{1}{\kappa\left(\frac{\|\hat{c}_{s'} - c_j\|}{h}\right)} \tag{1}$$

and $\sum_j w_j(\hat{c}_{s'}) = 1$. Here, $\mathcal{N}_K(\hat{c}_{s'})$ are indices of the k nearest neighbors of $\hat{c}_{s'}$ (under a chosen distance metric, e.g., Euclidean or cosine). Note that for constant kernel $\kappa\left(\|\hat{c}_{s'} - c_j\|/h\right) = \kappa$, the weights become uniform ($w_j = 1/k$), irrespective of distance. The reference pairs $\{(c_j, z_j)\}$ are derived from training data. Finally, the edited feature latents $\hat{z}_{s'}$ are decoded by the conditional generator f_θ to produce the new output frames: $\hat{x}_{s'} = f_\theta\left(\ell(\hat{c}_{s'}), y_{s'}\right)$. This methodology offers several distinct advantages over direct manipulation in a single latent space: First, by performing edits in the compact dynamics space $\mathcal{C}$, we gain fine-grained, stable, and interpretable control over sequence dynamics and condition-based modifications. Second, the use of the near-lossless feature latent space $\mathcal{Z}$ for final synthesis ensures that high image quality is maintained throughout the editing process. Third, the framework is capable of a range of tasks including reconstruction, interpolation and prediction, all achieved by manipulating the dynamics within the structured latent space $\mathcal{C}$.

In summary, our method leverages a dual-space architecture to decouple the challenging tasks of trajectory editing and image generation. We deliberately separate concerns: $\mathcal{C}$ is a compact, dynamics–aligned (lossy) embedding for robust trajectory modeling and control, while $\mathcal{Z}$ is a high–dimensional, near–lossless feature space for photorealistic generation. Neighborhood information in $\mathcal{C}$ (e.g., kNN) provides the auxiliary context needed to reconstruct $\mathcal{Z}$ for synthesis. This principled design enables stable and intuitive control over spatiotemporal sequences while preserving high generative fidelity. The formulation enables: *(1) Faithful interpolation/extrapolation* along trajectories in $\mathcal{C}$, including imputation of missing frames; *(2) Interpretable control* over condition changes $y_s \to y_{s'}$ via structured traversal in $\mathcal{C}$; and *(3) High–fidelity synthesis* by lifting edited embeddings to $\mathcal{Z}$ (via ℓ) and decoding with f_θ.

3.2 Encoding/Decoding to Feature Latent space via cLDM Framework.

In our work, we use conditional latent diffusion model (cLDM) to encode or decode to *Feature Latent space*, which operates diffusion process in the latent space $\mathcal{Z}$ of a pretrained VAE instead of pixel space $\mathcal{X}$ which is very high-dimensional. The forward diffusion process progressively adds Gaussian noise to the data and a reverse diffusion iteratively denoises it, using a U-Net architecture. (Rombach et al., 2022) demonstrated that this approach achieves state-of-the-art performance in tasks such as text-to-image synthesis while significantly reducing computational overhead. We denote $E : \mathcal{X} \to \mathcal{Z}$ as encoder and $D : \mathcal{Z} \to \mathcal{X}$ as decoder of the pretrained VAE. Based on image-conditioning pairs, we then learn the cLDM via

$$\mathcal{L}_{\text{cLDM}} := \mathbb{E}_{E(x),y,\,\epsilon\sim\mathcal{N}(0,1),\,t}\Big[\,\big\|\epsilon - \epsilon_\theta(z_t, t, y)\big\|_2^2\Big]. \tag{2}$$

Here, ϵ_θ denote the UNet model (Ronneberger et al., 2015), t is a diffusion timestep, z_t is a noisy version of the clean input $E(x)$, and $y \in \mathcal{Y}$ denotes conditional label. This reduces compute cost and helps the model focus on semantic features rather than pixel-level noise.

We employ DDIM inversion (Song et al., 2020; Dhariwal and Nichol, 2021) to invert a real encoded image $E(x)$ to a noisy latent code $z^\star = \tilde{g}_\phi(E(x), y)$ with $g_\phi(\cdot, y) = \tilde{g}_\phi(\cdot, y) \circ E$. When used as the starting point in the sampling process, this latent code enables reconstruction of the original image as $\hat{x} = (D \circ \tilde{f}_\theta)(z^\star, y) = f_\theta(z^\star, y)$ (see Appendix B.1 for details). In other words, DDIM inversion provides a deterministic mechanism to compute inverted latent $z^\star = z_T$, yielding a latent trajectory $\{z_t\}_{t=0}^T$ that maps cyclically to and from the data sample: This cycle-consistency property ensures that latent edits remain faithful when decoded back into data space.

3.3 Structured Embedding and Controlled Generation via ConDA Framework

The complete ConDA pipeline, including diffusion model training (or use of a pretrained model), diffusion inversion, contrastive embedding learning, trajectory traversal, kNN lifting, and diffusion sampling is summarized in Algorithm 1 (Training) and Algorithm 2 (Generation) in Appendix B.1.

Structured Embedding of Diffusion Latents. Although, the cLDM operates in compressed observation space, their DDIM-inverted latent space $\mathcal{Z}$ is still high-dimensional tensors and lacks priors to generate spatiotemporal data. Absent additional inductive biases, interpolations or edits need not correspond to smooth or meaningful changes in system dynamics. To address this, we introduce a supervised contrastive representation-learning framework ConDA and learn a map h_ψ that organizes $\mathcal{Z}$ into a low-dimensional space $\mathcal{C}$ that encodes ordered trajectories. Using a supervised InfoNCE objective, positives are defined as (z_i, z_p) belonging to the same label or condition $y_i = y_p$, while negatives are drawn from samples with differing labels. The supervised contrastive loss is given by

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}_{(i)}\left[\frac{1}{|P(i)|}\sum_{p\in P(i)}\log\frac{\exp\big(\text{sim}(c_i, c_p)/\tau\big)}{\sum_{a\neq i}\exp\big(\text{sim}(c_i, c_a)/\tau\big)}\right], \tag{3}$$

where $c_i = h_\psi(z_i, y_i)$, $P(i)$ denotes the set of positives for anchor i (same label), $\text{sim}(c_i, c_p) = -\|c_i - c_p\|_2^2$ is the similarity measure, and τ is a temperature parameter. This loss encourages embeddings with the same condition to cluster together while separating those with different conditions, enforcing that $\mathcal{C}$ encodes dynamics in a condition-aware, low-dimensional structure.

Trajectory Modeling and Controlled Generation. Within the space $\mathcal{C}$ we introduce approaches that enable nonlinear dynamic modeling and controlled generation. The goal is to identify directions of movement in latent space that correspond to meaningful changes in observed data, either along the trajectory of a sequence or across distinct classes. We define a map $\mathcal{T}_\eta : \mathcal{C} \times \mathcal{Y} \times \mathcal{Y} \to \mathcal{C}$ that shifts a source embedding (c_s, y_s) towards a target embedding $(c_{s'}, y_{s'})$, while leaving unrelated factors invariant. This is obtained by solving

$$\hat{c}_{s'} = \arg\min_{\mathcal{T}} \mathcal{L}\left(\mathcal{T}_\eta(c_s; y_s \to y_{s'}); c_{s'}\right) + \lambda\,\Omega\big(\mathcal{T}_\eta\big), \tag{4}$$

where $\mathcal{L}$ is a task-specific discrepency loss that ensures that the generated sample under the edited latent aligns with the target representation, while a regularization term $\Omega(\cdot)$, weighted by λ, enforces edits to be small. The solution, $\hat{c}_s$ thus reflects the minimal, structured change in the latent space

$\mathcal{C}$ needed to achieve faithful, controllable generation aligned with the target condition. This yields generated sample $\hat{x}_{s'} = f_\theta(l(\hat{c}_s), y_{s'})$ and end-to-end controlled generation framework:

$$x_s \xrightarrow{g_\theta} z_s \xrightarrow{h_\psi} c_s \xrightarrow{\mathcal{T}_\eta} \hat{c}_{s'} \xrightarrow{l} \hat{z}_{s'} \xrightarrow{f_\theta} \hat{x}_{s'}. \tag{5}$$

(1) Spline Interpolation. For sequential data, we assume that embeddings $c_{1:S}$ lie on or near a smooth manifold that captures the intrinsic phase of the sequence. To recover this structure, we fit a a C^2-continuous parametric curve $\gamma : \mathbb{R} \to \mathcal{C},\ \gamma(\alpha) \approx c_s$ for $\alpha = \alpha_s$, to the ordered pairs $\{(\alpha_s, c_s)\}_{s=1}^S$, where α_s is a monotone ordering parameter (e.g., normalized index or latent phase). Concretely, parametric spline curve γ is obtained by minimizing a regularized least-squares objective that balances data fidelity and curvature: $\min_\gamma \sum_{s=1}^S \|\gamma(\alpha_s) - c_s\|^2 + \lambda \int \|\gamma''(\alpha)\|^2 \, d\alpha$, subject to appropriate boundary conditions. This smooth parameterization lets us define edits as motion along the curve. Given a point c_s at coordinate α_s, the edit operator moves by a small phase increment $\Delta\alpha$:

$$\mathcal{T}(c_s;\, y_s \to y_{s+\Delta s}) \;=\; \gamma(\alpha_s + \Delta\alpha). \tag{6}$$

Choosing $\Delta\alpha > 0$ or $\Delta\alpha < 0$ traverses forward or backward along the trajectory, enabling interpolation of missing states and extrapolation to unobserved phases.

(2) Taylor Extrapolation (TEX). In this local extrapolation approach, we estimate the direction of change in $\mathcal{C}$ that moves a source embedding forward along the trajectory by a step size Δs using a second-order Taylor expansion around a local coordinate s:

$$\mathcal{T}(c_s; y_s \to y_{s+\Delta s}) = c(s + \Delta s) \;\approx\; c(s) \;+\; \dot{c}(s)\,\Delta s \;+\; \tfrac{1}{2}\ddot{c}(s)\,(\Delta s)^2, \tag{7}$$

where $\dot{c}, \ddot{c}$ are the first and second derivatives of the local trajectory estimated using finite differences. The first order TEX-1 uses only the linear term, while the second order TEX-2 adds $\frac{1}{2}\ddot{c}(s)\,(\Delta s)^2$, which captures local nonlinearities of the trajectory. This allows us to navigate the latent space along locally directed trajectories, enabling generation of sequences consistent with the estimated dynamics.

(3) Classification+KDE-Based Class Traversal. Given sequences of latent embeddings $\{\, c_{1:S}^{(j)} \,\}_{j=1}^N$, we first train an SVM with an RBF kernel on $\{c^{(j)}\}$ to separate classes (e.g., responder vs. non-responder). To identify navigation directions between classes, we estimate class-conditional densities via KDE: $f_k(u) = \frac{1}{N_k} \sum_{i=1}^{N_k} K\Big((u - c_i^{(k)})/h\Big),\ k \in \{0, 1\}$, where $K(\cdot)$ is a Gaussian kernel, h is the bandwidth, and $\{c_i^{(k)}\}$ are samples from class k. We then form the density difference $\Delta f(u) = f_1(u) - f_0(u)$, and detect class-specific peaks by $m_{\text{class0}} = \arg\max_u\big(-\Delta f(u)\big),\ m_{\text{class1}} = \arg\max_u\big(\Delta f(u)\big)$. The transformation function $\mathcal{T}$ here is defined as a traversal from the source to the target class along the line connecting these peaks:

$$\mathcal{T}_\eta(c_s^{(\text{class0})}; y_s^{\text{class0}} \to y_s^{\text{class1}}) = m_{\text{class0}} + \eta\big(m_{\text{class1}} - m_{\text{class0}}\big), \quad \eta \in [0, 1]. \tag{8}$$

This procedure provides interpretability by linking movements in latent space to density maxima of the two classes, enabling smooth and explainable transitions between non-responder and responders.

3.4 Running example: DISFA facial expressions

In DISFA Mavadati et al. (2013), each sample is a short video of one subject transitioning from a neutral to an expressive face and back (e.g., a smile involving AU12/25). We encode each frame x_s and its AU label y_s into a diffusion latent z_s via diffusion inversion (DDIM or Rex-RK4) of a pretrained conditional LDM, yielding a latent trajectory $(z_1, \ldots, z_S)$. ConDA then learns a low-dimensional embedding $c_s = h_\psi(z_s, y_s)$ such that distances and directions in the embedding space correspond to expression dynamics (e.g., position within the smile, AU intensity). At generation time, we draw a smooth curve in the embedding space (e.g., from "neutral" to "smile" and back), lift each point back to diffusion latents via kNN decoding, and render the corresponding frames with the cLDM decoder to obtain a controllable facial-expression video. See Appendix B.1.1 for a fully annotated version of this example.

4 Experiments

Datasets. We evaluate on five spatiotemporal domains: fluid dynamics, neural calcium imaging, facial expression modeling, therapeutic neurostimulation, and neural spike dynamics. Full acquisition/simulation details and splits are in Appendix A.

(1) **Flow past a cylinder.** 2D incompressible Navier–Stokes with cylinder interactions following the benchmark of Schäfer et al. (1996); simulated in FEniCS (Logg et al., 2012). We render n=246,000 RGB images (velocity magnitude) conditioned on time y=τ. To reduce temporal leakage, we use blocked splits (Roberts et al., 2017).

(2) **Two-photon calcium imaging (PFC).** High-resolution recordings from mouse prefrontal cortex during cognitive flexibility tasks (Spellman et al., 2021). We convert videos to n=706,452 RGB images with condition y=(v_n, v_t) (session and within-session frame).

(3) **DISFA facial expressions.** Standard AU dataset (Mavadati et al., 2013). We use n=261,576 frames with condition labels y=AU $\in \{0, \ldots, 5\}^{12}$ and perform contiguous 80/20 splits.

(4) **TMS-induced electric fields.** Individualized head models and E-field simulations with SimNIBS (Thielscher et al., 2015); n=569,520 images from 121 patients, conditioned on coil angle y=Θ. Used for classification and class-transfer analyses.

(5) **Monkey reach neural spiking.** Premotor population activity during delayed reaches (Churchland and Kaufman, 2022; Pei et al., 2021). Following Kapoor et al. (2024), spikes are mapped to 12-D S4 latents; we train conditional diffusion directly on these latents with 2D reach velocity as y.

Experimental setup. Complete hyperparameters and algorithmic steps appear in Appendix C.1 and Appendix B.1. For images, we use the LDM autoencoder (Rombach et al., 2022) to encode $x \in \mathbb{R}^{256\times256\times3}$ to $E(x)\in\mathbb{R}^{32\times32\times4}$, train a `UNet2DModel` (Hugging Face `diffusers` (Von Platen et al., 2022)) with 1000 diffusion steps, and apply DDIM inversion to obtain feature latents z. Contrastive embeddings are learned with CEBRA (Schneider et al., 2023) (`offset10-model-mse`); we use d=8 for fluid and d=3 otherwise (Appendix Fig. 10). We lift $c \in \mathbb{R}^d$ to latent space with a kNN decoder and render via DDIM sampling. For n-steps prediction ahead, the data are split using M contiguous train/test blocks, each containing n-length held-out points: Fluid: $M = 50,\ n = 12;\ Ca^{2+} : M = 100,\ n = 7;$ DISFA : $M = 88, n = 11$.

Tasks and metrics. Across datasets we evaluate three sequence-level tasks: (i) reconstruction and short-range interpolation of sequences; (ii) n-step-ahead prediction given a prefix; and (iii) class-transfer / counterfactual editing in the embedding space. Trajectory modeling uses parametric B-splines and finite-difference TEX on latent sequences $z_{1:S}$; baselines are linear (Lerp), spherical (Slerp) and an LSTM. Image fidelity is measured by PSNR and SSIM, and trajectory error by RMSE. *Trajectory modeling:* parametric B-splines and finite-difference TEX on sequences $z_{1:S}$; baselines are linear (Lerp), spherical (Slerp) (Wang and Golland, 2023; Hahm et al., 2024; Preechakul et al., 2022), and an LSTM (Hochreiter and Schmidhuber, 1997). *Image fidelity:* PSNR and SSIM; *trajectory error:* RMSE. *Classification:* linear/RBF SVMs with leave-subject-out (TMS) and leave-(Re)-out (flow) (Osafo Nkansah et al., 2024); we report F1/Acc/AUC. *Class transfer:* KDE interpolation between class-conditional modes in the embedding (details in Appendix C.1). *Non-image dynamics:* on monkey S4 latents, we compare ConDA embeddings to PCA in the same diffusion latents using RMSE, total absolute error (mean/sd), and Procrustes distance.

5 Results

We benchmarked our cLDM against a continuous-label conditioned GAN Ding et al. (2020), finding that cLDM significantly outperforms the GAN in both image generation and reconstruction. Appendix C.7 provides full protocols and detailed results. In this section, we present both qualitative and quantitative results with cLDM for our proposed approaches. In addition to assessing full-reference image quality using PSNR and SSIM, we evaluate *trajectory-level* accuracy by measuring the RMSE between predicted and ground-truth embeddings.

Nonlinear Trajectory Modeling: Spline/TEX-2 vs. Baselines. Our experiments reveal that the inherent, complex nonlinear dynamics within the high-dimensional $\mathcal{Z}$ space are difficult to model directly. This challenge is clearly demonstrated by the results in Table 1, where linear baselines consistently underperform, and even nonlinear baselines like LSTM and TEX-1 show limited accuracy when operating solely within $\mathcal{Z}$. However, the notably better performance scores (higher PSNR and SSIM, significantly lower RMSE) across all methods, particularly Spline and TEX-2, when applied in the lower-dimensional, contrastive space ($\mathcal{C}$) validate our approach. This evidence along with qualitative comparisons in Figure 2 and generalization results in Table 2, supports our core finding:

Data	Method	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓
		DDIM $\mathcal{Z}$-space			ConDA $\mathcal{C}$-space		
Fluid	Lerp	28.16 ± 0.49	0.56 ± 0.08	16.12	28.27 ± 0.47	0.59 ± 0.07	13.75
	Slerp	28.11 ± 0.53	0.54 ± 0.09	17.24	28.27 ± 0.48	0.59 ± 0.07	13.09
	LSTM	34.50 ± 1.81	0.92 ± 0.09	4.04	34.53 ± 2.05	0.92 ± 0.08	1.07
	TEX-1	29.28 ± 2.38	0.57 ± 0.25	14.20	32.98 ± 3.09	0.84 ± 0.15	3.16
	TEX-2	$\mathbf{35.29 \pm 0.60}$	$\mathbf{0.94 \pm 0.01}$	**0.26**	$\mathbf{35.70 \pm 0.36}$	$\mathbf{0.94 \pm 0.01}$	0.02
	Spline	—	—	—	$\mathbf{35.70 \pm 0.36}$	$\mathbf{0.94 \pm 0.01}$	**0.00**
Ca^{2+}	Lerp	32.36 ± 1.68	0.79 ± 0.04	16.60	32.92 ± 2.86	0.82 ± 0.04	43.87
	Slerp	32.68 ± 2.54	0.80 ± 0.07	17.65	32.82 ± 2.81	0.82 ± 0.04	43.38
	LSTM	35.52 ± 2.70	0.86 ± 0.07	9.15	35.96 ± 2.91	0.87 ± 0.07	0.89
	TEX-1	30.64 ± 1.07	0.57 ± 0.08	22.58	36.13 ± 2.96	0.87 ± 0.07	0.84
	TEX-2	$\mathbf{38.00 \pm 0.51}$	$\mathbf{0.92 \pm 0.01}$	**0.13**	$\mathbf{38.58 \pm 0.59}$	$\mathbf{0.93 \pm 0.01}$	**0.00**
	Spline	—	—	—	$\mathbf{38.58 \pm 0.59}$	$\mathbf{0.93 \pm 0.01}$	**0.00**
DISFA	Lerp	29.98 ± 2.70	0.74 ± 0.08	13.86	33.08 ± 3.25	0.83 ± 0.09	13.61
	Slerp	31.13 ± 2.72	0.76 ± 0.09	14.97	33.13 ± 3.31	0.83 ± 0.09	13.68
	LSTM	38.13 ± 0.54	$\mathbf{0.96 \pm 0.00}$	3.64	38.74 ± 0.75	0.96 ± 0.01	0.34
	TEX-1	35.18 ± 4.06	0.88 ± 0.14	7.49	38.77 ± 0.70	0.96 ± 0.00	0.24
	TEX-2	$\mathbf{38.27 \pm 0.24}$	$\mathbf{0.96 \pm 0.00}$	**0.07**	$\mathbf{38.99 \pm 0.23}$	$\mathbf{0.96 \pm 0.00}$	**0.00**
	Spline	—	—	—	$\mathbf{38.99 \pm 0.23}$	$\mathbf{0.96 \pm 0.00}$	**0.00**

Table 1: Baseline comparison of interpolations. We compare spline and TEX-2 (second-order Taylor expansion) against Lerp, Slerp, LSTM, and TEX-1 (first-order Taylor). Linear baselines (Lerp/Slerp) consistently underperform in both $\mathcal{Z}$ and $\mathcal{C}$, indicating inherently nonlinear trajectories. In $\mathcal{Z}$, TEX-2 yields the best fidelity. In $\mathcal{C}$, spline and TEX-2 perform best, consistent with a dynamics-aligned geometry that supports smooth, predictable traversal. Overall, nonlinear methods (LSTM, TEX) improve notably in $\mathcal{C}$ compared to $\mathcal{Z}$.

by learning contrastive embeddings, the $\mathcal{C}$ space effectively captures the underlying dynamics in a more structured, dynamics-aligned geometry. This transformation simplifies the problem by creating smoother, more stable trajectories that are amenable to modeling. The exceptional performance of our proposed Spline and TEX-2 methods in $\mathcal{C}$ (achieving near-zero RMSE) further underscores the advantages of this space for accurately representing and predicting these complex dynamics. Consequently, not only is interpolation and prediction significantly more accurate in $\mathcal{C}$, but the simplified and more stable nature of its trajectories also makes it a more suitable and effective space for direct editing and manipulation of the dynamics compared to the raw diffusion latent space ($\mathcal{Z}$).

Method	Fluid			Ca^{2+}			DISFA		
	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓
	ConDA $\mathcal{C}$-space			ConDA $\mathcal{C}$-space			ConDA $\mathcal{C}$-space		
LSTM	33.62±2.21	0.89±0.11	0.58	35.12±2.60	0.86±0.06	1.00	36.44±2.80	0.93±0.06	0.75
TEX-2	34.44±2.13	0.91±0.10	0.50	34.92±2.56	0.86±0.06	1.01	36.37±2.89	0.92±0.08	0.61
Spline	32.89±3.19	0.84±0.15	0.58	34.90±2.54	0.85±0.15	1.09	36.98±2.29	0.94±0.05	0.52

Table 2: n-step-ahead prediction using nonlinear extrapolation methods (Spline, TEX-2, and LSTM). The results show that generalization remains stable and is not degraded by approximation sources such as k-NN lifting, diffusion inversion, and VAE reconstruction, indicating that these errors are negligible for downstream prediction and do not impede faithful recovery or extrapolation.

SVM Classification and KDE-Based Class Interpolation. We evaluated the effectiveness of the proposed ConDA space ($\mathcal{C}$) for downstream tasks by training linear and RBF-kernel SVMs for binary classification on two datasets: Fluid (steady vs. unsteady laminar flow) and E-Field (responder vs. non-responder). As shown in Table 3, models trained on the low-dimensional $\mathcal{C}$-space consistently and significantly outperformed those trained directly in the high-dimensional diffusion latent space ($\mathcal{Z}$) on all metrics (Accuracy, F1, and ROC-AUC), regardless of the kernel used. This robust improvement highlights that the $\mathcal{C}$-space effectively organizes task-relevant information into a more separable and predictable manifold. For instance, in the E-Field task, the SVM-RBF classifier trained on $\mathcal{C}$ achieved an F1 score of 0.63, a marked improvement over the 0.30 achieved in $\mathcal{Z}$. Leveraging the interpretability afforded by the low dimensionality of $\mathcal{C}$, we visualized the decision-relevant latent dynamics. We used kernel density estimation to model the class-conditional densities and then interpolated along the vector connecting the peaks of these densities. The resulting sequences, depicted in Figure 3, illustrate the smooth trajectory of change required for a non-responder E-field

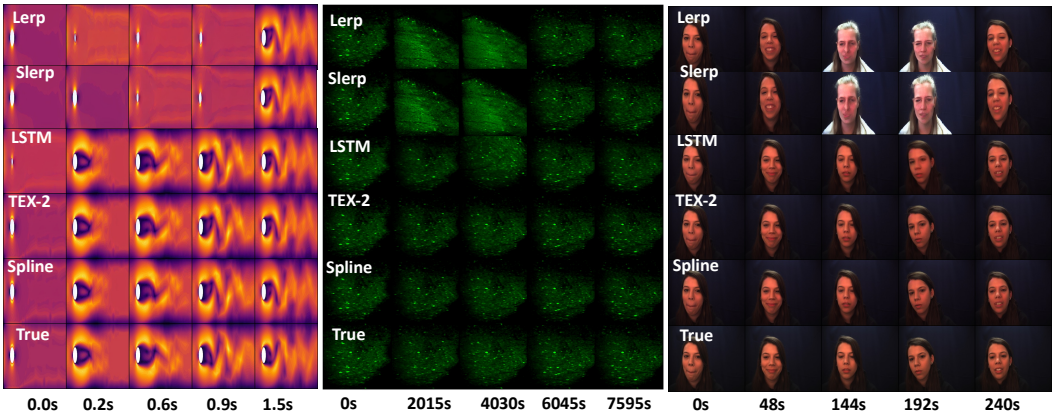


Figure 2: Visual comparison of generated images from interpolations in $\mathcal{C}$-space. Linear methods (Lerp, Slerp) show inferior perceptual quality, whereas Spline and TEX-2 yield smoother, more dynamics-consistent reconstructions.

Method	Acc. ↑	F1 ↑	AUC ↑
Steady vs. Unsteady Laminar Flow			
$\mathcal{Z}$-Space (D=4096)			
SVM-Linear	0.78	0.73	0.89
SVM-RBF	0.68	0.52	0.59
$\mathcal{C}$-Space (d=3)			
SVM-Linear	0.83	0.81	0.82
SVM-RBF	**0.87**	**0.85**	**0.95**
Responder vs. Non-responder E-Field			
$\mathcal{Z}$-Space (D=4096)			
SVM-Linear	0.48	0.21	0.46
SVM-RBF	0.57	0.30	0.58
$\mathcal{C}$-Space (d=3)			
SVM-Linear	0.62	0.62	0.68
SVM-RBF	**0.66**	**0.63**	**0.67**

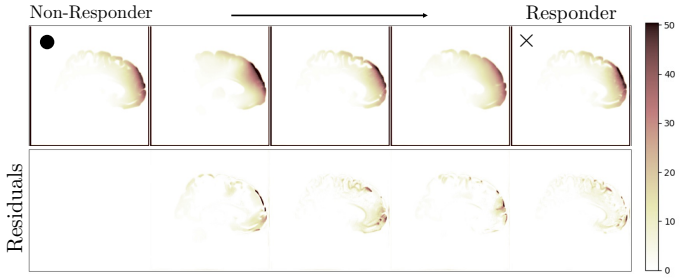


Figure 3: SVM classification performance and KDE based class interpolation. (Left) Metrics for classifying steady vs. unsteady laminar flow and E-field responder vs. non-responder using $\mathcal{Z}$ and $\mathcal{C}$-space. (Right) Trajectory morphing a non-responder E-Field (•) towards a responder (×) by interpolating between class-conditional KDE peaks; the color bar shows per-pixel E-field change from the source, highlighting cortical shifts needed for responder-like patterns and suggesting a path to personalized targeting.

pattern to morph toward that of a responder. The color-coded changes reveal specific cortical shifts, providing a tangible and interpretable path toward personalizing targeting strategies.

Ablation: ConDA vs. linear and variational baselines. We compared our ConDA method against two baseline representation learning techniques: PCA, a linear method, and a β-VAE ($\beta = 4$), a variational method for the task of test-sequence prediction. To ensure a fair comparison, all models used identical train/validation splits and were configured with the same latent dimension, d. Each representation was evaluated with its corresponding decoder. As summarized in Table 3, ConDA consistently achieved superior performance across all datasets, as measured by PSNR, SSIM, and RMSE. For example, on the DISFA dataset, ConDA shows a clear advantage in RMSE (5.50) and PSNR (36.26). Qualitative comparisons on DISFA (Figure 4) further illustrate ConDA's ability to recover fine facial motions at unseen frames with higher fidelity. These results indicate that the baseline models fail to capture the dynamics-relevant variance as effectively as ConDA.

Interpretability of ConDA-space $\mathcal{C}$. Setup. We evaluated interpretability on neural spiking data from a monkey performing a delayed center–out reach task (Kapoor et al., 2024). Our experimental setup followed the LDNS pipeline, using an autoencoder with S4 layers to map 182-channel spike trains to 12-dimensional, time-aligned latents. We then trained conditional diffusion models on these latents, conditioned on 2D hand velocity. For evaluation, we computed ConDA embeddings ($\mathcal{C}$) and PCA embeddings of the DDIM-inverted S4 latents. Alignment with ground-truth velocity and reach conditions was measured using standard probes, including RMSE, the mean/std of the

Method	PSNR ↑	SSIM ↑	RMSE ↓
Fluid			
PCA	31.82 ± 2.30	0.87 ± 0.11	10.43
β-VAE	32.20 ± 2.55	0.85 ± 0.14	**7.47**
ConDA	**33.90 ± 2.52**	**0.90 ± 0.11**	7.60
Ca^{2+}			
PCA	34.73 ± 2.29	0.86 ± 0.05	12.40
β-VAE	34.62 ± 2.40	0.85 ± 0.06	11.58
ConDA	**35.69 ± 2.54**	**0.87 ± 0.05**	**11.02**
DISFA			
PCA	34.82 ± 1.03	0.92 ± 0.02	7.60
β-VAE	35.76 ± 1.23	0.94 ± 0.02	6.59
ConDA	**36.26 ± 1.03**	0.94 ± 0.01	**5.50**

Table 3: Baseline comparison: ConDA vs linear and variational baselines. Performance metrics for test sequence prediction comparing ConDA to PCA and β-VAE. ConDA achieves improved performance indicating closer alignment with the underlying dynamical manifold.

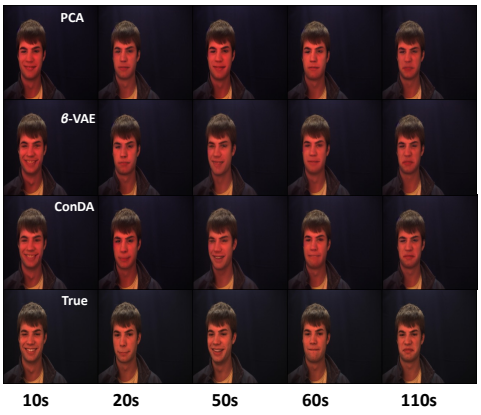
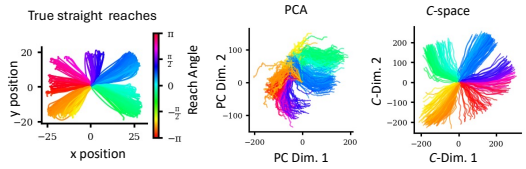


Figure 4: Comparison of facial expressions at test frames. ConDA captures facial expressions (e.g., lip-corner pull/stretch) at test frames with higher fidelity than PCA and β-VAE.



Metric	PCA	$\mathcal{C}$-Space
RMSE ↓	1.71 ± 0.29	$\mathbf{1.46 \pm 0.45}$
TAE mean/std ↓	5.56	**3.08**
Procrustes Distance ↓	0.83 ± 0.16	$\mathbf{0.73 \pm 0.22}$

Figure 5: Interpretability of $\mathcal{C}$ on monkey reach spiking data. Using LDNS Kapoor et al. (2024) with structured state-space (S4) layers, we map neural spikes to time-aligned latents and train diffusion models conditioned on velocity/reach. Comparing 2-dimensional ConDA embedding $\mathcal{C}$ and PCA embeddings of neural spikes to ground-truth velocity conditions, $\mathcal{C}$ space aligns more closely, indicating higher interpretability.

total absolute error (TAE), and geometric alignment metrics such as Procrustes distance. We then assess alignment with the ground-truth velocity/reach conditions using standard probes: RMSE, mean/std of total absoulte error, and geometric alignment metrics (e.g., Procrustes fit). As illustrated in Fig. 5, embeddings in $\mathcal{C}$ exhibit clearer organization by reach direction and stronger alignment with true velocity trajectories than PCA. The performance metrics consistently favor $\mathcal{C}$ over PCA, indicating that contrastive structuring yields representations whose local geometry better reflects task-relevant kinematics. This improves interpretability relative to variance-only PCA embeddings. Additional evidence of the near-orthogonality of the latent subspace directions associated with temporal variations and those influenced by confounding factors (such as the Reynolds number in the fluid dataset) is provided as evidence of successful disentanglement (Appendix C.6).

6 Conclusion and Limitations

We introduced ConDA, a contrastive alignment layer that organizes diffusion latents into compact, dynamics-aware embeddings, enabling smooth nonlinear traversals that outperform raw latent and linear baselines across physical and biological domains. ConDA is solver-agnostic and operates consistently across different inversion solvers (e.g., DDIM, Rex-RK4 Blasingame and Liu (2025)) as well as across different diffusion models, including Isometric Diffusion Hahm et al. (2024) latents (see Appendix C). The method has three **limitations**: (1) the embedding is lossy and many-to-one, which we mitigate using a neighborhood-preserving kNN lifting decoder (with learned decoders as future work); (2) the embedding emphasizes local rather than global geometry, which may distort long-range traversals, though locally fitted interpolators and models such as Isometric Diffusion can help; and (3) the approach relies on DDIM inversion, adding computational cost that can be reduced through caching, multi-scale pipelines, or faster solvers such as Rex-RK4 or consistency models. Despite these trade-offs, approximation errors remain small in practice, and ConDA provides a practical path toward controllable and interpretable modeling of complex dynamical systems.

REFERENCES

Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4432–4441, 2019.

Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18208–18218, 2022.

Dmitry Baranchuk, Ivan Rubachev, Andrey Voynov, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Zander W Blasingame and Chen Liu. A reversible solver for diffusion sdes. *arXiv preprint arXiv:2502.08834*, 2025.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18392–18402, 2023.

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15): 3932–3937, 2016.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

M Churchland and M Kaufman. Mc_maze: macaque primary motor and dorsal premotor cortex spiking activity during delayed reaching. *Data set*, 2022.

Eleanor J Cole, Katy H Stimpson, Brandon S Bentzley, Merve Gulser, Kirsten Cherian, Claudia Tischler, Romina Nejad, Heather Pankow, Elizabeth Choi, Haley Aaron, et al. Stanford accelerated intelligent neuromodulation therapy for treatment-resistant depression. *American Journal of Psychiatry*, 177(8):716–726, 2020.

Jingyi Cui, Weiran Huang, Yifei Wang, and Yisen Wang. Aggnce: Asymptotically identifiable contrastive learning. In *NeurIPS Workshop*, 2022.

Mohammad Daneshzand, Sergey N Makarov, Lucia I Navarro de Lara, Bastien Guerin, Jennifer McNab, Bruce R Rosen, Matti S Hämäläinen, Tommi Raij, and Aapo Nummenmaa. Rapid computation of tms-induced e-fields using a dipole-based magnetic stimulation profile approach. *Neuroimage*, 237:118097, 2021.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Xin Ding, Yongwei Wang, Zuheng Xu, William J Welch, and Z Jane Wang. Ccgan: continuous conditional generative adversarial networks for image generation. In *International Conference on Learning Representations*, 2020.

Maria Drakaki, Claus Mathiesen, Hartwig R Siebner, Kristoffer Madsen, and Axel Thielscher. Database of 25 validated coil models for electric field simulations for tms. *Brain Stimulation*, 15(3):697–706, 2022.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487, 2020.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2022.

Jaehoon Hahm, Junho Lee, Sunghyun Kim, and Joonseok Lee. Isometric representation learning for disentangled latent space of diffusion models. *arXiv preprint arXiv:2407.11451*, 2024.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022a.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022b.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Jaivardhan Kapoor, Auguste Schulz, Julius Vetter, Felix Pei, Richard Gao, and Jakob H Macke. Latent diffusion for neural spiking data. *Advances in Neural Information Processing Systems*, 37:118119–118154, 2024.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

Patrick Kidger, James Foster, Xuechen Chen Li, and Terry Lyons. Efficient and accurate gradients for neural sdes. *Advances in Neural Information Processing Systems*, 34:18747–18761, 2021a.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6696–6707, 2021b.

D Kinga, Jimmy Ba Adam, et al. A method for stochastic optimization. In *International conference on learning representations (ICLR)*, page 6. San Diego, California;, 2015.

Runjin Liu, Yitong Li, Kevin Swersky, Yiren Zhang, David J Fleet, and Raquel Urtasun. Diffusion models for sequential data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*. Springer Science & Business Media, 2012.

Qi Lyu and Xiao Fu. On finite-sample identifiability of contrastive learning-based nonlinear independent component analysis. In *International Conference on Machine Learning*, pages 14582–14600. PMLR, 2022.

S Mohammad Mavadati, Mohammad H Mahoor, Kevin Bartlett, Philip Trinh, and Jeffrey F Cohn. Disfa: A spontaneous facial action intensity database. *IEEE Transactions on Affective Computing*, 4(2):151–160, 2013.

Sam McCallum and James Foster. Efficient, accurate and stable gradients for neural odes. *arXiv preprint arXiv:2410.11648*, 2024.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

PC Miranda, L Correia, R Salvador, and PJ Basser. Tissue heterogeneity as a mechanism for localized neural stimulation by applied electric fields. *Physics in Medicine & Biology*, 52(18):5603, 2007.

Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Alexander Opitz, Mirko Windhoff, Robin M Heidemann, Robert Turner, and Axel Thielscher. How the brain tissue shapes the electric field induced by transcranial magnetic stimulation. *Neuroimage*, 58(3):849–859, 2011.

Isaac Osafo Nkansah, Neil Gallagher, Ruchi Sandilya, Conor Liston, and Logan Grosenick. Generalizing cnns to graphs with learnable neighborhood quantization. *Advances in neural information processing systems*, 37: 82318–82349, 2024.

Felix Pei, Joel Ye, David Zoltowski, Anqi Wu, Raeed H Chowdhury, Hansem Sohn, Joseph E O'Doherty, Krishna V Shenoy, Matthew T Kaufman, Mark Churchland, et al. Neural latents benchmark'21: Evaluating latent variable models of neural population activity. *arXiv preprint arXiv:2109.04463*, 2021.

Ben Poole, Sherjil Ozair, Aäron van den Oord, Alexander A Alemi, and George Tucker. On variational lower bounds of mutual information. In *NeurIPS Workshop on Bayesian Deep Learning*, 2018.

Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10619–10629, 2022.

Oula Puonti, Koen Van Leemput, Guilherme B Saturnino, Hartwig R Siebner, Kristoffer H Madsen, and Axel Thielscher. Accurate and robust whole-head segmentation from magnetic resonance images for individualized head modeling. *Neuroimage*, 219:117044, 2020.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Kashif Rasul, Chris Seward, Irina Schuster, Uwe Bergmann, Arthur Gretton, and Siamak Ravanbakhsh. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning (ICML)*, 2021.

David R Roberts, Volker Bahn, Simone Ciuti, Mark S Boyce, Jane Elith, Gurutzeta Guillera-Arroita, Severin Hauenstein, José J Lahoz-Monfort, Boris Schröder, Wilfried Thuiller, et al. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography*, 40(8):913–929, 2017.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

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.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent odes for irregularly-sampled time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Michael Schäfer, Stefan Turek, Franz Durst, Egon Krause, and Rolf Rannacher. *Benchmark computations of laminar flow around a cylinder*. Springer, 1996.

Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, 2023.

Siming Shan, Pengkai Wang, Song Chen, Jiaxu Liu, Chao Xu, and Shengze Cai. Pird: Physics-informed residual diffusion for flow field reconstruction. *arXiv preprint arXiv:2404.08412*, 2024.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020.

Zexuan Shen et al. Interpretable counterfactual explanations for deep image classification by concept-based interaction. In *arXiv preprint arXiv:2206.01022*, 2022.

Yuguang Shi, Yitong Chen, Yang Song, Hongyu Ren, Yixuan Luo, Diederik Kingma, and Stefano Ermon. Sequence modeling with diffusion processes. In *International Conference on Learning Representations (ICLR)*, 2021.

Dule Shu, Zijie Li, and Amir Barati Farimani. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 2023.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

Timothy Spellman, Malka Svei, Jesse Kaminsky, Gabriela Manzano-Nieves, and Conor Liston. Prefrontal deep projection neurons enable cognitive flexibility via persistent feedback monitoring. *Cell*, 184(10):2750–2766, 2021.

Axel Thielscher, Alexander Opitz, and Mirko Windhoff. Impact of the gyral geometry on the electric field induced by transcranial magnetic stimulation. *Neuroimage*, 54(1):234–243, 2011.

Axel Thielscher, Andre Antunes, and Guilherme B Saturnino. Field modeling for transcranial magnetic stimulation: a useful tool to understand the physiological effects of tms? In *2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 222–225. IEEE, 2015.

Vikram Voleti, Alexia Jolicoeur-Martineau, and Chris Pal. Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in neural information processing systems*, 35:23371–23385, 2022.

Patrick Von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models, 2022.

Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22532–22541, 2023.

Clinton J Wang and Polina Golland. Interpolating between images with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Fangyikang Wang, Hubery Yin, Yue-Jiang Dong, Huminhao Zhu, Hanbin Zhao, Hui Qian, Chen Li, et al. Belm: Bidirectional explicit linear multi-step sampler for exact inversion in diffusion models. *Advances in Neural Information Processing Systems*, 37:46118–46159, 2024.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pages 9929–9939. PMLR, 2020.

Chen Henry Wu and Fernando De la Torre. A latent space of stochastic diffusion models for zero-shot image editing and guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7378–7387, 2023.

Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3121–3138, 2022.

Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. *arXiv preprint arXiv:2403.01742*, 2024.

Guoqiang Zhang, Jonathan P Lewis, and W Bastiaan Kleijn. Exact diffusion inversion via bidirectional integration approximation. In *European Conference on Computer Vision*, pages 19–36. Springer, 2024.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

Juntang Zhuang, Nicha C Dvornek, Sekhar Tatikonda, and James S Duncan. Mali: A memory efficient and reverse accurate integrator for neural odes. *arXiv preprint arXiv:2102.04668*, 2021.

APPENDIX FOR NONLINEARLY CONTROLLABLE COUNTERFACTUALS WITH DIFFUSION

DATASETS AND CODES ACCESS INFORMATION

All code used to produce results presented in this work will be released as open source under the MIT license. We also make our Flow Past a Cylinder benchmarking dataset publicly available to serve as a physically complex video data set with ground truth; it can be found here [Box link will be added; currently removed for anonymization], and full details on how it was produced are below. The two-photon imaging of Ca2+-sensor data are available upon request from the authors of the original paper (Spellman et al., 2021). Details to produce the E-field data are below, but we cannot provide the HAMD or real neuroimaging data for patients as these data are part of an ongoing clinical trial and are HIPAA-protected.

A DATA ACQUISITION

A.1 SIMULATION OF FLOW PAST A CIRCULAR CYLINDER

The flow past a cylinder is a fundamental fluid dynamics problem with numerous practical applications in engineering, science, and industry. As the Reynolds number Re increases, the interesting nonlinear phenomenon of Karman vortex shedding occurs and the flow becomes time-periodic with vortices shedding behind the cylinder. For low Reynolds numbers, the flow remains stationary.

For flow around a circular cylinder, a two-dimensional model is sufficient to capture the essential flow features. The underlying flow geometry and boundary conditions are illustrated in Figure 6. Assuming a fluid density of $\rho = 1.0$, the fluid dynamics are governed by the non-stationary Navier-Stokes equations:

$$\mathbf{u}_t - \nu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = 0, \quad \nabla\cdot\mathbf{u} = 0$$

where $\mathbf{u}$ represents the velocity and p the pressure. Here the kinematic viscosity is set to $\nu = 0.001$. No-slip boundary conditions are applied to the lower and upper walls, as well as the boundary of the cylinder. On the left edge, a parabolic inflow profile is prescribed:

$$\mathbf{u}(0, y) = \left(\frac{4Uy(0.41 - y)}{0.41^2}, 0\right)$$

with a maximum velocity $U = \frac{3\nu Re}{4r}$, where Re and r denote the Reynolds number and the radius of the cylinder, respectively. On the right edge, do-nothing boundary conditions define the outflow:

$$\nu\frac{\partial\mathbf{u}}{\partial n} - p\mathbf{n} = 0$$

with $\mathbf{n}$ representing the outer normal vector.

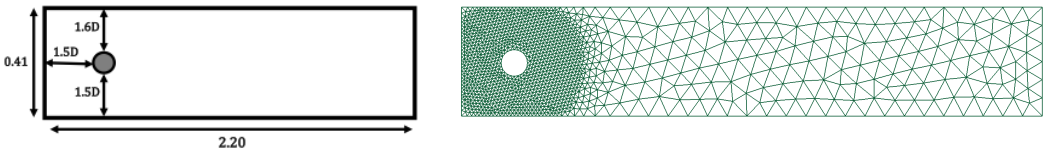
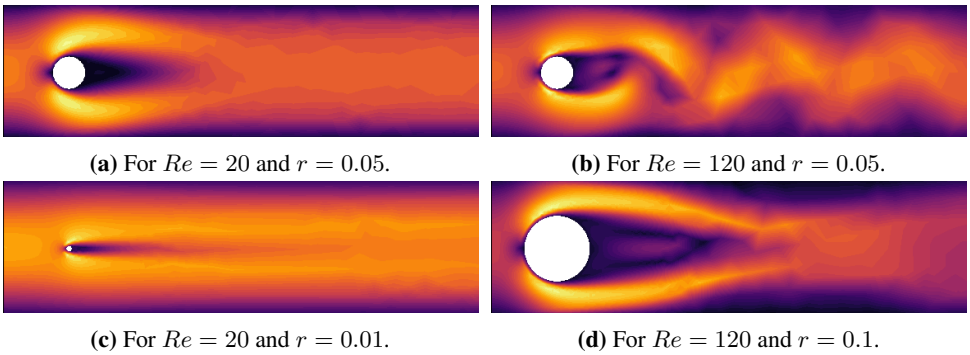


Figure 6: (Left) The geometry utilized in fluid simulations, envisioned as a pipe without a cylindrical structure with diameter $D = 2r$. (Right) An adaptively refined mesh with 1446 nodes and 2897 element for fluid flow simulations.

We use FEniCS Logg et al. (2012), a Finite Element Method (FEM) library, to solve the governing Navier-Stokes equations on an adaptively refined mesh for spatial discretization as shown in Figure 6. Our simulations generate a dataset with four groups exhibiting different flow behaviors based on their input parameters: the first group has Reynolds numbers (Re) ranging from 20 to 40 with a cylinder radius (r) of 0.05 meters; the second group has Re ranging from 100 to 120 with $r = 0.05$ meters; the third group has r ranging from 0.01 to 0.05 meters with $Re = 20$; and the fourth group has r ranging from 0.05 to 0.1 meters with $Re = 120$. The simulations cover a time range from 0 to



(a) For $Re = 20$ and $r = 0.05$. **(b)** For $Re = 120$ and $r = 0.05$.

(c) For $Re = 20$ and $r = 0.01$. **(d)** For $Re = 120$ and $r = 0.1$.

Figure 7: Diverse flow characteristics observed across various flow groups at time 1.5 secs. Note the aspect ratio was changed to yield square images for the analyses in the main text (allowing comparisons with CcGAN, which requires square images).

1.5 seconds with a time step size of 0.005 seconds, capturing the persistent behavior of the system, whether characterized by stationary or periodic states.

Transforming Fluid Flow Simulations into Image Space. After obtaining the FEM solution for the flow past a circular cylinder, we use the plot function from DOLFIN, which is based on Python libraries such as Matplotlib, to visualize the velocity magnitude at a given time step. This function converts floating-point simulation data into an RGBA image representation. Additional customization and saving of the plots as image files are handled using Matplotlib, resulting in a dataset of 256×256 pixel images.

A.2 Simulation of TMS-induced Electric Field

Since the early stages of Transcranial Magnetic Stimulation (TMS), field calculations have been employed for designing coils and, more recently, for assessing the spatial stimulation pattern induced by TMS stimulation in the brain. These calculations are vital for enhancing the precision, effectiveness, and safety of TMS interventions, and they serve a central role in advancing our understanding of the neural mechanisms behind TMS efficacy and in developing personalized treatment strategies.

We use a volume conductor model to simulate the electric field distribution in a patient's brain during TMS. Such a model factors in tissue conductivity, individualized head anatomy, and TMS coil parameters, optimizing predictions based on variations in skull thickness and tissue boundaries. Rooted in Maxwell's equations, the model accounts for the coil's size, shape, and orientation, as well as stimulation parameters, influencing the strength and characteristics of the induced electric field. Most of the tools that are available for realistic field calculations rely on methods such as FEM and head models that accurately capture the important anatomical features. We use the SimNIBS library Thielscher et al. (2015) for head model reconstruction and for personalized E-Field simulations. We first create a subject-specific tetrahedral head mesh from T1-weighted and T2-weighted magnetic resonance (MR) scans using a bash script "charm" Puonti et al. (2020). The generation of the head model is the most time-consuming step and takes ≈ 3 hrs. The final meshes contain around $612,278$ nodes and $3,610,350$ tetrahedra (see Fig. 8), divided into different tissue classes. After the generation of a volume head model, FEM solvers are used to calculate the cortical distribution of electric fields in response to different TMS coil positions, intensities, and orientations.

TMS employs time-varying magnetic fields (B-fields) to induce electric fields (E-fields) according to Maxwell-Faraday law Thielscher et al. (2011); Opitz et al. (2011); Daneshzand et al. (2021) $\nabla \times \mathbf{E} = \frac{\partial \mathbf{B}}{\partial t}$. The total E-Field has the general form $\mathbf{E} = \frac{\partial \mathbf{A}}{\partial t} - \nabla\varphi$, where the first term $\frac{\partial \mathbf{A}}{\partial t}$ involving the magnetic vector potential $\mathbf{A}$ corresponds to the primary field (excitation) induced by the current in the coil, whereas the term $\nabla\varphi$ involving the scalar potential φ is called the secondary field. The primary and secondary fields are coupled through the condition of volumetric quasi-neutrality $\nabla \cdot \mathbf{J} = \nabla \cdot (\sigma \mathbf{E})$ where $\mathbf{J}$ denotes the current density and σ denotes the electric conductivity of the tissue. The secondary field is generated by charge accumulation at the conductivity boundaries to render the normal component of the current continuous Miranda et al. (2007). The FEM solver implements the Galerkin method based on tetrahedral first order elements to determine φ at the nodes.

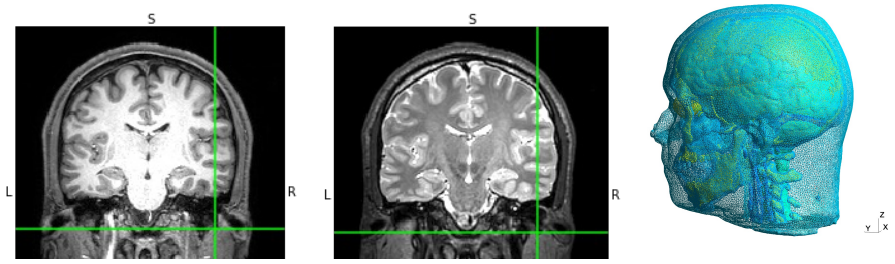


Figure 8: Illustration of T1w (left) and T2w (center) images of a representative subject from SimNIBS used for head model reconstruction. The right panel shows the detailed, subject-specific head mesh used for personalized E-Field simulations.

The electric fields and current densities are determined in each mesh element based on the above equations. One simulation takes around 6 minutes and uses a maximum of $\sim$4 GB memory.

Our simulations aim to generate TMS induced E-field datasets of depressed patients with the coil centered at their dorsolateral prefrontal cortex (DLPFC) coordinates with different coil angles. The goal is to study the performance of our method in generalizing the distributions of E-Field over different coil angles and different brain regions in real time. We consider simulating E-Fields mapped to NIfTI volume slices which accounts for each patient's unique functional neuroanatomy and cortical folding patterns. The coil is centered at the DLPFC coordinate of an individual pointing posteriorly towards different angles $y = \Theta \in (0°, 360°)$ with angle resolution $\Delta\Theta = 10°$ within a circle formed by considering gray matter vertices within a distance of 20mm and clustering the vertices into $360°/\Delta\Theta$ clusters using k-means.

For model training, we consider T1w and T2w images of $n = 121$ treatment resistant depressed patients undergoing accelerated trials at Weill Cornell Medicine, Cornell University for head model reconstruction. We use their DLPFC coordinates as the coil target and consider coil orientations determined by the $360°/\Delta\Theta$ different angles. The coil model used was Magventure Cool-B65 and therefore we set stimulation intensity to $80A/\mu s$ Drakaki et al. (2022) consistent with TMS treatment levels. The coil-to-cortex distance was set to 4 mm.

Transforming E-Field Simulations into Image Space. The outcomes of E-Field simulations are saved in both Gmsh and NIfTI formats, with the latter mapping the E-Field values onto subject-specific volumes. To process this data, we utilize the NiBabel Python package, which allows us to access the E-Field values mapped onto NIfTI volumes as NumPy arrays. These arrays match the orientation and dimensions of 3D brain imaging data. For visualization, we generate 2D images by slicing through the 3D imaging data volume, with each slice representing a distinct cross-sectional view of the subject at a precise location. Using Matplotlib, we then save these visualized E-Field mappings as RGBA image files. Through this method, we effectively convert E-Field simulation outputs into image data (256×256 pixels).

Identifying as Responders or Non-Responders. In our nonlinear classification analysis, to classify patients as responders or non-responders we analyzed the HAMD-17 scores of $n = 110$ subjects for which we had pre- and post-treatment HAMD-17 scores and who received TMS treatment targeted to DLPFC, calculating the percentage change from baseline using the formula:

$$\frac{\text{HAMD-17}_{\text{after}} - \text{HAMD-17}_{\text{before}}}{\text{HAMD-17}_{\text{before}}} \times 100,$$

where $\text{HAMD-17}_{\text{before}}$ and $\text{HAMD-17}_{\text{after}}$ represent the scores before and after treatment, respectively. Patients who exhibited a decrease of 50% or more in their HAMD-17 scores were classified as responders, while those with less than a 50% reduction were classified as non-responders (consistent with prior work Cole et al. (2020)), see Figure 9.

A.3 2-Photon Calcium Imaging Datasets of Mouse PFC

Tracking the activity of specific neuronal populations in the prefrontal cortex (PFC) involved in cognitive flexibility provides important insights into the neural mechanisms underlying such behavior.

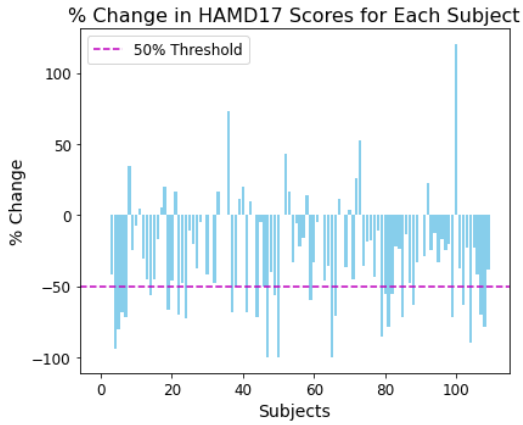


Figure 9: Percent change in HAMD-17 scores. Dashed line represents the threshold to classify a patient as a treatment responder, defined as a percent change of $\leq 50\%$. Based on this threshold, 30 patients are identified as treatment responders, while 80 patients are identified as non-responders

In this work, we use a comprehensive dataset from Spellman et al. (2021), which includes high-resolution calcium imaging data capturing PFC neuronal dynamics in mice as they perform behavioral tasks that require flexible cognitive control.

Two-photon laser scanning microscopy was employed to image pyramidal neurons expressing the genetically encoded calcium indicator GCaMP6f. The imaging field of view was designed to preserve cortical laminar structure and included both the prelimbic and infralimbic regions of the PFC. Calcium signals were acquired at a resolution of 256×130 pixels, spanning a $1500, \mu$m $\times 760, \mu$m area, corresponding to a spatial resolution of $5.85, \mu$m per pixel. Each scan lasted 346 milliseconds, yielding a frame rate of 2.89 Hz. The dataset comprises video recordings from 21 mice, with approximately three imaging sessions per animal and a total of 1,462 behavioral trials. To enable cross-session analysis of the same neuronal populations, non-rigid co-registration was performed using Python package CellReg.

The behavioral paradigm involved a structured sequence of task transitions designed to assess different aspects of cognitive flexibility. These included: simple discrimination (SD), where mice discriminated between two stimuli within a single sensory modality; compound discrimination (CD), where an irrelevant stimulus from a different modality was added; intradimensional shift (IDS), which introduced new exemplars within the same modality; reversal (Rev), where the stimulus-response mapping was switched; extradimensional shift (EDS), where the relevant modality was changed for the first time; a second IDS (IDS2), introducing new exemplars within the newly relevant modality; and serial extradimensional shift (SEDS), where the task rule switched automatically upon reaching performance criterion. Throughout this paradigm, task-related neural activity was recorded using two-photon calcium imaging through a coronally implanted microprism, allowing simultaneous visualization of prelimbic and infralimbic areas while preserving laminar architecture. Further methodological details are provided in Spellman et al. (2021).

To train the conditional Latent Diffusion Model (cLDM), the calcium imaging data were transformed into RGB images of resolution 256×256 pixels. Each image was conditioned on a frame identifier $v_n \in 0, 1, \ldots, 59$ and a trial time point $v_t \in 0, 1, \ldots, 19999$, forming the conditioning variable $y = (v_n, v_t)$. In total, 706,452 images were generated and used for training and validating the cLDM model.

A.4 DISFA DATASET

As a facial expression dynamics dataset, we use DISFA dataset Mavadati et al. (2013), a widely used benchmark in affective computing, facial action unit (AU) recognition, and temporal emotion analysis. It consists of high-resolution spontaneous facial expression videos recorded from 27 subjects while they watched video stimuli designed to elicit natural emotional responses. Each video sequence is annotated frame-by-frame with Facial Action Units (AUs) following the Facial Action Coding System

(FACS).DISFA contains approximately 4,844 frames (recorded at 20 frames per second) for each subject, each with intensity scores for 12 AUs: AU1 (inner brow raise), AU2 (outer brow raise), AU4 (brow lowerer), AU5 (upper lid raise), AU6 (cheek raise), AU9 (nose wrinkler), AU12 (lip corner puller), AU15 (lip corner depressor), AU17 (chin raiser), AU20 (lip stretcher), AU25 (lips part), and AU26 (jaw drop). Intensities are discretized on a 6-point ordinal scale from $\{0, 1, 2, 3, 4, 5\}$ where 0 indicates absence and 5 denotes maximum intensity. For our experiments, we transform video frames into 261,576 RGB images (from both left and right video camera) and pair each frame with its AU intensity vector, giving condition labels $y = \text{AU} \in \{0, 1, 2, 3, 4, 5\}^{12}$.

B METHOD

Algorithm 1 Diffusion Model Training + Diffusion Inversion + ConDA Training

Inputs: Training data $\{(x_s, y_s)\}$, noise schedule α_t, pretrained VAE encoder E, distance metric $d(\cdot, \cdot)$, temperature τ, positive window Δ
Outputs: Trained diffusion model ϵ_θ, diffusion latents z_T, trained contrastive encoder h_ψ

Stage A: Diffusion Model Training (ϵ_θ) (e.g., cLDM, IsoDiff, AE-s4; model may be pre-trained)

Initialize $z_0 = E(x_s)$
repeat
  Sample $z_0 \sim p(E(x) \mid y)$
  Sample $t \sim \text{Uniform}\{1, \ldots, T\}$
  Sample $\epsilon \sim \mathcal{N}(0, I)$
  Compute $z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$
  Take gradient step on $\|\epsilon - \epsilon_\theta(z_t, t, y_s)\|^2$
until convergence
Fix UNet ϵ_θ

Stage B: Diffusion Inversion (g_θ) (e.g., DDIM, Rex-RK4)

Initialize $z_0 = E(x_s)$
for $t = 1, \ldots, T$ **do**
  DDIM inversion: $z_t = \dfrac{\sqrt{\alpha_t}\,(z_{t-1} - \sqrt{1 - \alpha_{t-1}}\,\epsilon_\theta(z_{t-1}, t, y_s))}{\sqrt{\alpha_{t-1}}} + \sqrt{1 - \alpha_t}\,\epsilon_\theta(z_{t-1}, t, y_s)$
end for
Store inverted latent $z_T = z_s$

Stage C: ConDA (our contribution) (h_ψ)

Build training tuples $\{(z_s, y_s)\}$ by sampling frames
repeat
  Construct positive set: $P(i) = \{(z_j, y_j) : d(y_j, y_i) \le \Delta\}$ and negative set: $N(i) =$ others
  Compute embeddings $c_s = h_\psi(z_s, y_s)$ for all items in a minibatch
  Optimize the InfoNCE objective:

$$\mathcal{L}_{\text{InfoNCE}} = -\sum_i \frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(\text{sim}(c_i, c_p)/\tau)}{\sum_{q \in P(i) \cup N(i)} \exp(\text{sim}(c_i, c_q)/\tau)}$$

  where $\text{sim}(u, v) = -\|u - v\|^2$
until validation loss saturates
Tune embedding dimension via cross-validation on reconstruction scores
Fix encoder $\hat{h}_\psi = h_\psi$

Algorithm 2 Prediction/Traversal + kNN Lifting + Diffusion Sampling (Generation)

Inputs: Latent trajectory $(c_1, \ldots, c_s)$, neighbor count $k^\star$, trained DDIM sampler f_θ
Outputs: Predicted/edited trajectory $c_{s'}$, generated sample $\hat{x}$

Stage A: Prediction / Traversal

Use traversal approach $\mathcal{T}$ (Spline, TEX-2, etc.)
Estimate local derivatives: $\dot{c}(s) \approx \frac{c_s - c_{s-1}}{\Delta s}, \quad \ddot{c}(s) \approx \frac{c_s - 2c_{s-1} + c_{s-2}}{(\Delta s)^2}$
Predict next latent: $c(s + \Delta s) \approx c(s) + \dot{c}(s)\Delta s + \frac{1}{2}\ddot{c}(s)(\Delta s)^2$
Set $c_{s+1} \leftarrow \mathcal{T}(c_s)$
Repeat recursively for multi-step prediction

Stage B: kNN Lifting (l)

Build training pairs (c_i, z_i)
For held-out c, compute neighbor set: $N_{k^\star}(c) = \{i_1, \ldots, i_{k^\star}\}$
Compute weighted latent estimate: $l(c) = \dfrac{\sum_{i \in N_{k^\star}(c)} w_i z_i}{\sum_{i \in N_{k^\star}(c)} w_i}, \quad w_i = \dfrac{1}{d(c, c_i) + \varepsilon}$
Select $k^\star$ via cross-validation over grid $\mathcal{K} = k_1, k_2, \ldots$ using latent-space R^2 score
Fix $k^\star$ for inference
Lift predicted embedding: $l(c_{s'}) = z_{s'}$

Stage C: Sampling (f_θ) (e.g., DDIM, Rex-RK4)

Set $z_T = z_{s'}$
for $t = T, \ldots, 1$ **do**
    DDIM Sampling: $z_{t-1} = \dfrac{\sqrt{\alpha_{t-1}}(z_t - \sqrt{1 - \alpha_t}\,\epsilon_\theta(z_t, t, y_{s'}))}{\sqrt{\alpha_t}} + \sqrt{1 - \alpha_{t-1}}\,\epsilon_\theta(z_t, t, y_{s'})$
end for
Return z_0
Decode: $\hat{x} = D(z_0)$

B.1 Algorithmic Pipeline of the ConDA Framework

In this section, we present **Algorithm 1 (Training)** and **Algorithm 2 (Generation)**, which explicitly formalize the full ConDA pipeline:

1. *cLDM training → diffusion inversion → ConDA training*, and
2. *Prediction/Traversal → k-NN lifting → diffusion sampling*.

B.1.1 Running example of ConDA with DISFA facial expression sequence

In DISFA, each sample is a short video of one subject transitioning from a neutral face to an expressive face and back (e.g., a smile associated with AU12/25 activation; 12 AUs total). Each video consists of a frame sequence $(x_1, \ldots, x_S)$ and corresponding per-frame AU labels $(y_1, \ldots, y_S)$.

We encode every frame x_s into a diffusion latent z_s using DDIM inversion (g_θ) (Algorithm 1, Stage B) applied to a pretrained latent diffusion model (trained using Algorithm 1, Stage A), treating $\{z_s\}_{s=1}^S$ as the subject's latent trajectory. ConDA then learns a low-dimensional embedding $c_s = h_\psi(z_s, y_s)$ in $\mathcal{C}$-space (Algorithm 1, Stage C) such that geometric distance and direction in $\mathcal{C}$-space align with expression dynamics (e.g., time within the expression cycle or AU intensity). In practice, this means that moving along a simple one-dimensional curve in $\mathcal{C}$-space corresponds to smoothly increasing or decreasing the strength of a smile.

For generation, we sample a trajectory in $\mathcal{C}$-space (e.g., a linear or spline path from "neutral" to "expressive" and back) (Algorithm 2, Stage A), map it back to diffusion latents via the kNN lifting operator (l) (Algorithm 2, Stage B), and decode each latent using diffusion sampling (f_θ) to obtain a

controllable facial-expression video (Algorithm 2, Stage C). We evaluate predicted trajectories using RMSE in latent space, and evaluate generated frames using PSNR and SSIM.

B.2 CcGAN Inversion: Encoder Training

For a given generator function $G(\mathbf{z}, y) \sim f_\theta(\mathbf{z}, y)$ of the CcGAN, we train the encoder $E_z(\mathbf{x})$ by minimizing the following loss function:

$$\mathcal{L}_{ez} = \mathbb{E}_{\mathbf{z}\sim p_z,(\mathbf{x},y)\sim p_{data}}\Big[\underbrace{\|\mathbf{x} - G(E_z(\mathbf{x}), y)\|_2^2}_{\mathcal{L}_{ez_1}} + \eta \underbrace{\|\mathbf{z} - E_z(G(\mathbf{z}, y))\|_2^2}_{\mathcal{L}_{ez_2}}\Big]. \tag{9}$$

Here, $\mathcal{L}ez_1$ represents the squared reconstruction loss in image space, and $\mathcal{L}ez_2$ is the squared cyclic loss in the latent space. Note that the first objective both minimizes the difference between the input image $\mathbf{x}$ and its reconstruction $G(E_z(\mathbf{x}), y)$ and second objective seeks to minimize cyclic loss between the latent code $\mathbf{z}$ and its round-trip projection through the data space and back to the latent spaces via $E_z(G(\mathbf{z}, y)$. These two objectives are traded off by tunable hyperparameter $\eta \in \mathbb{R}_+$.

C Results

C.1 Experimental Setup

cLDM Framework. For cLDM training, we utilize the pre-trained VAE from Rombach et al. (2022) and `UNet2DModel` from Hugging Face diffusers library Von Platen et al. (2022). This VAE produces a smaller representation of an input image and then reconstructs the image based on this small latent representation with a high degree of fidelity. It takes in 3-channel images and produces a 4-channel latent representation with a reduction factor of 8 for each spatial dimension. That is, a $3 \times 256 \times 256$ input image will be compressed down to a $4 \times 32 \times 32$ latent. The U-Net model is used to take noisy latent samples, timesteps, and conditional labels as inputs to predict noise in the diffusion process. We utilize a linear multistep noise scheduler from diffusers library Von Platen et al. (2022) to introduce noise with diffusion steps $T = 1000$ for training the model and for inference. The model is trained for $40,000$ iterations using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 64. For inversion, we apply DDIM inversion with $T = 1000$ steps. As a solver baseline, we also implement the reversible Rex-RK4 method Blasingame and Liu (2025) using the noise-prediction parameterization, coupling parameter 0.9999, and 8 steps. We evaluate Rex-RK4 with both cLDM and the pretrained Isometric Diffusion model Hahm et al. (2024).

Contrastive learning and controlled generation. To obtain contrastive embeddings of the sequence of feature latents, we use the CEBRA library Schneider et al. (2023) with the `offset10-model-mse` architecture. Embedding dimensions (d) are tuned (Figure 10), with $d = 8$ for fluid and $d = 3$ for all other datasets, ensuring $d << \dim(\mathcal{Z})$. Embeddings $c_s \in \mathbb{R}^d$ are mapped back to feature-latent space using `cebra.KNNDecoder` followed by image reconstruction via a diffusion sampler. Trajectory accuracy is measured using RMSE, while full-reference image quality is evaluated using PSNR and SSIM. The number of neighbors in the kNN decoder is tuned to maximize prediction accuracy.

For trajectory modeling, we use parametric B-spline interpolation (`splprep`/`splev` from SciPy, suitable for $d \leq 10$) and TEX (via finite-difference). Experiments are conducted on sequences from: fluid ($Re = 120,\ S = 3000$), Ca^{2+} ($v_n = 9,\ S = 6200$) and DISFA (subject = SN025, $S = 4844$). We compare against linear baselines, linear interpolation (Lerp) and spherical interpolation (Slerp) Hahm et al. (2024); Preechakul et al. (2022); Wang and Golland (2023), and a nonlinear baseline, an LSTM Hochreiter and Schmidhuber (1997) predicting the next frame from the previous 20 frames. For n-steps prediction ahead, the data are split using M contiguous train/test blocks, each containing n-length held-out points: Fluid: $M = 50,\ n = 12$; $Ca^{2+} : M = 100,\ n = 7$; $DISFA : M = 88, n = 11$. We evaluate spline extrapolation, TEX-2, and LSTM forecasting the next n frames from the previous 20 frames. As an ablation, we compared ConDA against linear (PCA) and variational baselines (β-VAE with $\beta = 4$) on predicting the test data.

For classification, we train linear and RBF SVMs with leave-subject-out CV on the TMS E-Field dataset with 110 subjects(responders: $\leq 50\%$ change in HAMD-17; non-responders: $> 50\%$ change in HAMD-17 shown in Figure 9) and leave-(Re)-out on the fluid dataset (steady: $Re \leq 65$; unsteady

laminar: $Re > 65$) Osafo Nkansah et al. (2024)). Generalization is evaluated on 10 held-out subjects and 8 held-out Re reporting F1, accuracy, and ROC-AUC. For class transfer, we perform KDE-based interpolation (`sklearn.neighbors.KernelDensity`) in five steps between non-responder and responder E-fields at a fixed coil angle $y = 90°$ by locating class-conditional density modes (`skimage.feature.peak-local-max`) and editing embeddings along the vector connecting mode peaks.

CcGAN Framework. We also benchmarked our cLDM against CcGAN with soft vicinity Ding et al. (2020); Miyato and Koyama (2018) for image generation and reconstruction. After significant testing, we use $40,000$ training iterations using the Adam optimizer Kinga et al. (2015) with β_1 set to 0.5, β_2 set to 0.999, dimension of latent space of GAN set to 256, learning rate set to 10^{-5} and batch size set to 64. Once the CcGAN is trained, we train an encoder network for the generator to enable inversion. The encoder network architecture, comprised of eight layers of $Conv2d \to BatchNorm2d \to ReLU$, had a bottleneck dimension of 256. The encoder was trained for $40,000$ iterations by minimizing the loss function defined in equation 9 with $\eta = 0.1$, using Adam optimizer with learning rate 10^{-5} and batch size 32.

Computational Resources. We conducted all training and evaluation using NVIDIA RTX 6000 GPUs with 24GB of memory. The number of trainable parameters for each model is summarized in Table 5, and the runtime for each simulation is reported in Table 6. The evaluation runtime was measured using a batch size of 64 samples.

Component	Used in	Pretrained	Trained	Objective (Loss)
VAE	cLDM	Yes	No	KL div.+ Reconst. loss (MSE)
U-Net	cLDM	No	Yes	Noise Pred. (MSE)
IsoDiff	Diffusion Baseline	Yes	No	Noise Pred. (MSE) + Isometry Loss
CEBRA	ConDA	No	Yes	InfoNCE
SVM	Classify	No	Yes	Hinge loss
β-VAE	$\mathcal{C}$ Baseline	No	Yes	Reconst. loss (MSE) + $\beta\times$ KL Divergence
LSTM	Prediction	No	Yes	Seq. regression (MSE)
MLP	$\mathcal{C}$ lifting Baseline	No	Yes	MSE

Table 4: Summary of model components with their pretraining, training, and fine-tuning stages, along with the corresponding loss functions.

Model	Model Size
CcGAN	77.48M
IsoDiff	113.67M
VAE	83.65M
UNet	113.68M
CEBRA	6.56M
β-VAE ($\mathcal{Z}$-Space)	18.89M
LSTM ($\mathcal{Z}$-Space)	13.65M
LSTM ($\mathcal{C}$-Space)	0.20M
MLP ($\mathcal{C}$-Space)	1.12M

Table 5: Trainable parameters.

Method	Fluid	E-Field	2P Ca^{2+}	DISFA
FEM Simul.	$\sim$70 hrs	$\sim$18 days	–	–
CcGAN Train.	$\sim$34 hrs	$\sim$43 hrs	–	–
CcGAN Eval.	$\sim$28 sec	$\sim$28 sec	–	–
cLDM Train.	$\sim$72 hrs	$\sim$73 hrs	$\sim$57 hrs	$\sim$87 hrs
cLDM Eval.	$\sim$5 min	$\sim$5 min	$\sim$5 min	$\sim$5 min
DDIM	2.31 s/samp.	–	2.31s/samp	1.29 s/samp.
Rex RK4	0.10 s/samp.	–	0.18 s/samp.	0.09 s/samp.

Table 6: Methods and runtime.

C.2 Diffusion Inversion Solvers: Rex-RK4 vs. DDIM reconstruction

Diffusion-model inversion is inherently numerically unstable, as small perturbations from floating-point noise, discretization, or batching can amplify during the reverse trajectory, a phenomenon well-documented in neural differential equations Kidger (2022). Recent work on exact diffusion inversion aims to eliminate this drift under low NFE budgets typical of modern pipelines, spanning ODE-based Blasingame and Liu (2025); Wallace et al. (2023); Zhang et al. (2024); Wang et al. (2024) and SDE-based approaches Blasingame and Liu (2025); Wu and De la Torre (2023), though many rely on heuristic, low-order schemes or require caching large parts of the forward trajectory, leaving their mathematical footing limited. A more principled direction develops algebraically reversible integrators, including MALI Zhuang et al. (2021), reversible adjoint methods Kidger et al. (2021a), higher-order reversible ODE solvers McCallum and Foster (2024), and reversible diffusion

ODE/SDE solvers Blasingame and Liu (2025), which offer improved stability. This context motivates our comparison of DDIM, a standard non-reversible first-order sampler, with Rex-RK4, a modern reversible Runge–Kutta solver, to test whether our method is solver-agnostic and whether reversibility improves inversion stability at the same NFE budget.

We compare DDIM inversion with the reversible Rex-RK4 integrator to assess reconstruction fidelity and evaluate solver dependence. As shown in Table 7, both methods closely match the VAE reconstruction. Importantly, Rex-RK4 achieves this performance with only 8 steps, providing a significantly lower NFE cost compared to DDIM (1000 steps) while maintaining near identical reconstruction fidelity.

Data	Method	PSNR ↑	SSIM ↑
Fluid	VAE	35.87 ± 0.38	0.95 ± 0.01
	Rex-RK4	35.91 ± 0.38	0.95 ± 0.01
	DDIM	35.94 ± 0.59	0.95 ± 0.01
Ca2+	VAE	38.61 ± 0.42	0.93 ± 0.01
	Rex-RK4	38.64 ± 0.41	0.93 ± 0.01
	DDIM	38.63 ± 0.42	0.93 ± 0.01
DISFA	VAE	39.07 ± 0.27	0.96 ± 0.00
	Rex-RK4	39.13 ± 0.25	0.96 ± 0.00
	DDIM	39.05 ± 0.27	0.96 ± 0.00

Table 7: Reconstruction performance of DDIM and Rex-RK4 (8 steps), with the VAE reconstruction defining the upper bound. Apparent 0.02–0.07 dB PSNR differences in favor of DDIM or Rex-RK4 arise from numerical noise (floating-point, batching, discretization) rather than genuine improvements. These results confirm that inversion errors are negligible relative to VAE error and do not impact on downstream outcomes.

C.3 Trajectory modeling: cLDM (Rex-RK4) and Isometric diffusion (Rex-RK4)

We compare interpolation approaches using an alternative diffusion inversion method, Rex-RK4, applied to both cLDM and the Isometric Diffusion model. The results in Tables 8 and 9 show that counterfactual behavior remains consistent across DDIM (see Table 1) and Rex-RK4 (the latter being much faster with only 8 steps), as well as when using Isometric diffusion (IsoDiff). IsoDiff model learns an isometric latent space directly within the diffusion model; we treat its latents as an additional source and apply ConDA on top of them. IsoDiff alone yields strong interpolation performance in its native latent space. Overall, these results confirm that ConDA does not depend on a particular inversion procedure or diffusion model.

Method	Fluid			Ca^{2+}			DISFA		
	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓
	cLDM Rex RK4 $\mathcal{C}$-space			cLDM Rex RK4 $\mathcal{C}$-space			cLDM Rex RK4 $\mathcal{C}$-space		
Lerp	28.75 ± 0.85	0.66 ± 0.08	11.90	37.13 ± 0.64	0.89 ± 0.01	34.40	34.97 ± 0.70	0.87 ± 0.02	23.26
Slerp	28.87 ± 0.92	0.67 ± 0.08	12.44	36.76 ± 0.71	0.89 ± 0.01	55.31	35.04 ± 0.77	0.87 ± 0.02	24.01
LSTM	34.13 ± 2.31	0.92 ± 0.04	0.66	38.28 ± 0.52	0.92 ± 0.01	1.83	38.71 ± 0.78	0.96 ± 0.01	0.51
TEX-1	33.35 ± 2.95	0.88 ± 0.09	3.17	38.24 ± 0.55	0.92 ± 0.01	2.09	38.68 ± 1.26	0.96 ± 0.01	0.73
TEX-2	**35.57 ± 0.34**	**0.94 ± 0.01**	0.01	**38.60 ± 0.58**	**0.93 ± 0.01**	**0.00**	**39.07 ± 0.23**	**0.96 ± 0.00**	**0.00**
Spline	**35.57 ± 0.34**	**0.94 ± 0.01**	**0.00**	**38.60 ± 0.58**	**0.93 ± 0.01**	**0.00**	**39.07 ± 0.23**	**0.96 ± 0.00**	**0.00**

Table 8: Baseline comparison of interpolations in $\mathcal{C}$-space using cLDM with Rex-RK4 (8 steps).

Data	Method	PSNR ↑	SSIM ↑	RMSE ↓	PSNR ↑	SSIM ↑	RMSE ↓
		IsoDiff Rex RK4 $\mathcal{Z}$-space			IsoDiff Rex RK4 $\mathcal{C}$-space		
DISFA	Lerp	35.14 ± 0.67	0.89 ± 0.02	21.49	35.11 ± 1.24	0.89 ± 0.04	31.45
	Slerp	35.35 ± 0.65	0.89 ± 0.02	21.51	35.46 ± 1.85	0.90 ± 0.04	35.25
	LSTM	38.75 ± 1.58	0.97 ± 0.02	7.53	38.30 ± 1.39	0.97 ± 0.02	1.29
	TEX-1	38.70 ± 1.46	0.97 ± 0.02	9.75	38.31 ± 2.27	0.97 ± 0.03	4.80
	TEX-2	**42.51 ± 0.02**	**1.00 ± 0.00**	**0.00**	**42.51 ± 0.02**	**1.00 ± 0.00**	0.01
	Spline	—	—	—	**42.51 ± 0.02**	**1.00 ± 0.00**	**0.00**

Table 9: Baseline comparison of interpolations in $\mathcal{Z}$ and $\mathcal{C}$-space using Isometric diffusion model from Hahm et al. (2024) with Rex-RK4 (8 steps) on DISFA facial expression datasets.

C.4 CONDA DECODERS: KNN VS MLP LIFTING

As a baseline, we compare kNN with an MLP decoder in Table 10 that takes a d-dimensional contrastive embedding and passes it through two fully connected layers with 256 units and ReLU activations (with optional dropout), followed by a final linear layer that maps to a 4096-dimensional output corresponding to the diffusion latent.

Data	Decoder	PSNR ↑	SSIM ↑
Fluid	MLP	30.10 ± 1.62	0.78 ± 0.11
	k-NN	**33.90 ± 2.52**	**0.90 ± 0.11**
Ca^{2+}	MLP	32.05 ± 1.02	0.79 ± 0.05
	k-NN	**35.01 ± 2.55**	**0.86 ± 0.06**
DISFA	MLP	34.08 ± 2.76	0.87 ± 0.06
	k-NN	**36.77 ± 2.58**	**0.94 ± 0.06**

Table 10: Comparison of k-NN (which leverages neighbor-based contrastive structure) and MLP lifting from $\mathcal{C}$-space to $\mathcal{Z}$-space. The results show that k-NN consistently outperforms a tuned MLP decoder across all datasets.

C.5 EFFECT OF CONTRASTIVE EMBEDDING SPACE DIMENSIONALITY ON TEST SEQUENCE PREDICTION

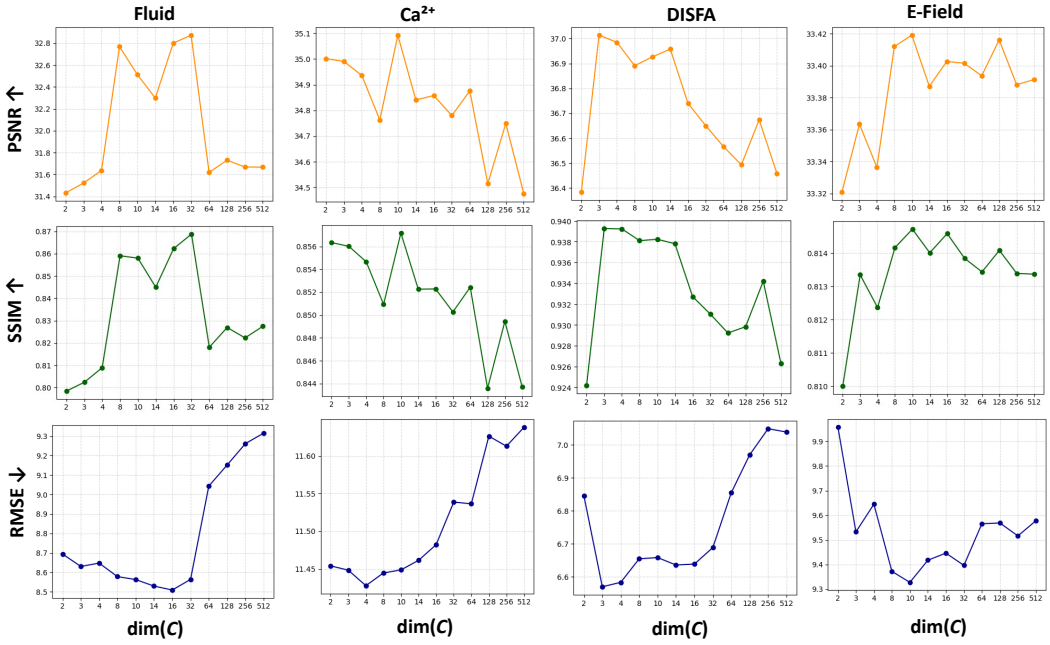


Figure 10: Prediction performance with embedding dimensionality. Test-set prediction across $\dim(\mathcal{C}) \in \{2, 3, 4, \cdots, 512\}$ shows negligible degradation for low-dimensional embeddings ($2 < \dim(\mathcal{C}) \leq 10$), indicating minimal information loss.

C.6 ORTHOGONALITY OF CONDA EMBEDDING SPACE

As a key benefit of InfoNCE-based contrastive learning, our ConDA embeddings create a latent space where distinct factors of variation are more orthogonal. To quantify disentanglement, we trained two separate linear regressions to predict the time (t) and Reynolds number (Re) from the latent embeddings. β_t and β_{Re} denote the learned regression coefficient vectors in each respective space. The cosine similarity between these two vectors is reported, with lower values indicating greater orthogonality. This is clearly demonstrated in Table 11, which reports the cosine similarity between the latent regression coefficients for time (β_t) and the confounding Reynolds number (β_{Re}) in the fluid dataset. Our ConDA space ($\mathcal{C}$) demonstrates significantly greater orthogonality between these latent

directions compared to the raw diffusion latent space ($\mathcal{Z}$), confirming that it better separates temporal variations from confounding physical parameters. This improved separation of factors—temporal variations from physical parameters—is crucial for fine-grained control, enabling users to edit the temporal evolution of the system while precisely controlling other physical attributes.

Latent Space	**Cosine Similarity** (β_t, β_{Re})
$\mathcal{Z}$ (dim = 4096)	0.1014
$\mathcal{C}$ (dim = 3)	0.0155

Table 11: Orthogonality of $\mathcal{C}$. Cosine similarity between latent regression coefficients for time and Reynolds numbers (β_t, β_{Re}) in fluid dataset. Lower values indicate greater orthogonality.

C.7 CcGAN vs. cLDM: Generated and Reconstructed Sample Comparison

To assess the quality and diversity of samples generated by CcGAN and cLDM, we performed both qualitative and quantitative analyses on two regression-conditioned datasets: fluid flow conditioned on time ($y = \tau$) and TMS-induced E-field distributions conditioned on coil angles ($y = \Theta$). We provide the visual comparison of generated samples in Fig. 11.

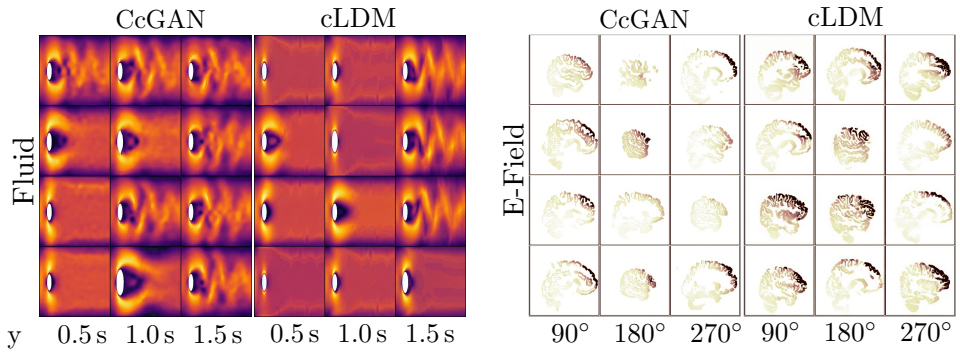
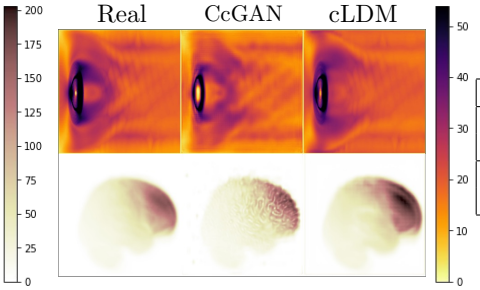


Figure 11: Visual comparison of randomly sampled images generated by CcGAN and cLDM. (Left) Generated fluid flow data at different time. **(Right)** Generated E-Field distributions at different TMS coil angles. Rsults indicate cLDM generates high quality samples.

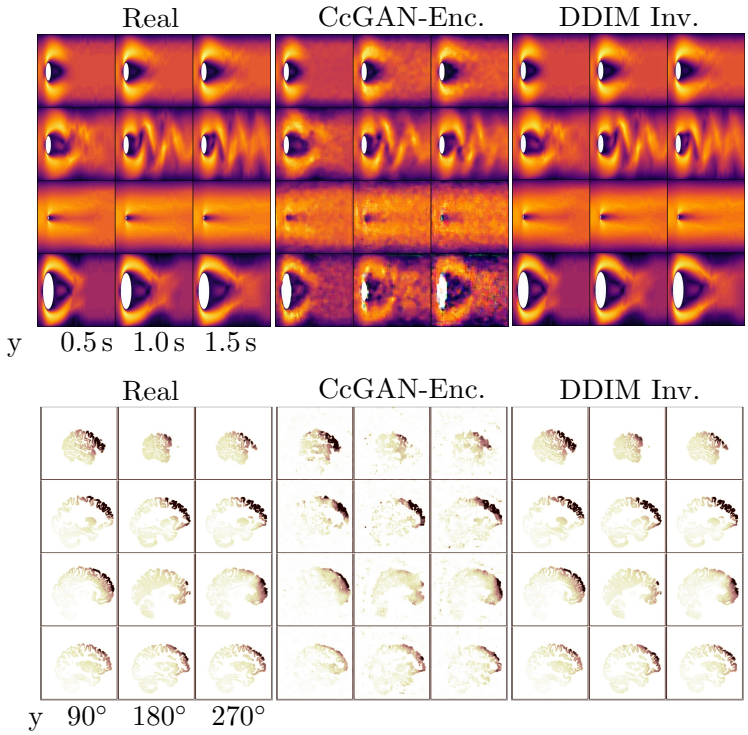
In Fig. 12, the **left** panel presents the standard deviation maps across 60k real and generated images to visualize spatial variability and structural complexity. The cLDM generated samples closely matched the variability of real data, effectively capturing diverse patterns such as varying cylinder radii in fluid flow and heterogeneous cortical foldings in E-Field maps. In contrast, CcGAN exhibited signs of mode collapse, failing to represent key variations, particularly smaller cylinder sizes and patient-specific cortical structures. For quantitative evaluation, we used Fréchet Inception Distance (FID) and Inception Score (IS) to measure image quality and diversity. As summarized in the Table of Fig. 12, cLDM achieved a lower FID and higher IS across both domains, indicating its ability to generate high-quality images with a distribution more consistent with the real data.

Fig. 13 compares reconstructed samples obtained from the CcGAN encoder and DDIM inversion. The top and middle panels display visual reconstructions, while the table in bottom panel summarizes quantitative evaluation using both general-purpose metrics (FID and IS) and domain-specific metrics (Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM)). These results highlight the improved reconstruction fidelity and better preservation of spatial and semantic structure offered by DDIM-based methods. Overall, cLDM reconstructions yield a more accurate and controllable representation of complex physical systems, supporting their suitability for nonlinear counterfactual generation tasks.



Dataset	Method	FID ↓	IS ↑
Fluid	CcGAN	0.670 ± 0.068	1.279 ± 0.019
	cLDM	**0.469** ± 0.089	**1.851** ± 0.023
E-Field	CcGAN	1.920 ± 0.790	1.435 ± 0.033
	cLDM	**1.246** ± 0.019	**2.083** ± 0.032

Figure 12: (Left) Visual comparison of standard deviation maps computed for each pixel across the sets of 60k real and generated samples with fluid and E-Field data, highlighting the variability and complexity in both sets. Note that CcGAN shows signs of mode collapse (e.g., the cylinder lacks size diversity and E-Field lacks diversity in unique cortical foldings across different patients). Scale bars are normalized velocity (a.u.) and normalized E-Field intensity (a.u.). **(Right)** Quantitative evaluation with lower FID and higher IS for cLDM generated samples indicate better diversity and quality than CcGAN.



Dataset	Method	FID ↓	IS ↑	PSNR ↑	SSIM ↑
Fluid	CcGAN-Enc.	0.947 ± 0.137	1.458 ± 0.079	30.946 ± 0.387	0.794 ± 0.047
	DDIM Inv.	**0.222** ± **0.017**	1.458 ± 0.048	**35.791** ± **0.229**	**0.954** ± **0.004**
E-Field	CcGAN-Enc.	1.383 ± 0.067	1.349 ± 0.069	34.296 ± 0.207	0.798 ± 0.013
	DDIM Inv.	**1.274** ± **0.040**	**1.745** ± **0.076**	**36.737** ± **0.149**	**0.955** ± **0.002**

Figure 13: Comparison of reconstructed images using CcGAN encoder and DDIM inversion. Visual assessment of reconstructed samples for two datasets: Different flow behavior conditioned over time **(top)** and TMS induced E-Field distributions conditioned on coil angles **(middle)**. **(Bottom)** Quantitative evaluation for reconstruction performance. Both visual and quantitative results demonstrate superior reconstruction quality using the DDIM-based inversion from the diffusion model.

C.8 SUPPLEMENTARY VIDEOS

Supplementary Video 1 presents synthetic video frames generated using the estimated latents $\hat{\mathbf{c}}$ from ConDA spline approach to model fluid dynamics at Re=120. The video highlights realistic vortex shedding behavior.

Supplementary Video 2 presents synthetic video frames generated using the ConDA TEX-2 approach to model 2P Ca^{2+} signal dynamics from the mouse PFC while performing a cognitive task. The video captures both realistic calcium activity dynamics and brain motion.

Supplementary Video 3 presents synthetic video frames generated using the ConDA prediction of subject's facial expression dynamics at test frames from the DISFA dataset.