IDENTIFYING ODES FROM UNSTRUCTURED DATA WITH CAUSAL REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the problem of recovering the governing ODE of a dynamical system from unstructured, high-dimensional observations such as images. Existing methods for ODE discovery typically assume direct measurements of the variables, or do not provide theoretical guarantees on the learned variables and equations. While Causal Representation Learning (CRL) methods provide guarantees on identifying variables from high-dimensional observations up to component-wise diffeomorphisms, we show that in general these variables cannot be used directly as input to equation discovery methods, which typically assume that the variables will lead to sparse equations. So we introduce SParse Equivalent Equation Discovery AutoEncoder (SPEED-AE), a framework that combines a pretrained CRL method with a component-wise autoencoder that learns transformations of variables that are amenable to sparse ODE discovery. We show that for polynomial ODEs, this additional step allows us to restrict the identifiability of each variable from polynomial to monomial diffeomorphisms. Experiments on Lotka-Volterra, Lorenz, and a two-pendulum system show that SPEED-AE improves on the disentanglement of the CRL methods and that it recovers ODEs that are closest to the ground truth, while achieving state-of-the-art forecasting performance.

1 INTRODUCTION

Machine learning is increasingly used for scientific discovery, especially for dynamical systems and Ordinary Differential Equations (ODEs). Much of this work targets prediction and forecasting aspects of physical systems (Chen et al., 2018; Hochreiter & Schmidhuber, 1997; Vaswani et al., 2017), possibly using ODEs as a tool to model the unknown dynamics of the system (Chen et al., 2018; Dupont et al., 2019; Heinonen et al., 2018; Rubanova et al., 2019). Equation discovery, i.e., learning ODE systems from data, has been extensively studied when direct access to the variables is available, both from a theoretical (Scholl et al., 2024; Yao et al., 2024) and a practical point of view (Brunton et al., 2016; Kaheman et al., 2020; Chen et al., 2024). Most results focus on identifiability in linear ODEs from single trajectories (Stanhope et al., 2014; Qiu et al., 2022), sparse coefficients (Casolo et al., 2025), and discrete observations (Wang et al., 2024b). By contrast, theoretical identifiability in non-linear ODEs remains limited to known functional forms (Grewal & Glover, 1976; Stanhope et al., 2014) or single initial conditions (Yao et al., 2024).

Few works address settings where variables are not directly observed, and only high-dimensional measurements (e.g., images or videos) are available. Existing approaches either learn equations on latent variables without guarantees that these variables correspond to the ground-truth (Champion et al., 2019; Auzina et al., 2023), or assume the underlying functional form of the ODE is known (Linial et al., 2021) and focus on parameter identification.

In this work, we provide theoretical guarantees for identifying the true variables from unstructured data up to simple indeterminacies, on which we then learn ODE equations that correspond up to similar indeterminancies to the ground truth ones. We build on Causal Representation Learning (CRL) (Schölkopf et al., 2021) methods, which provide guarantees on identifying variables from high-dimensional observations up to permutations and component-wise diffeomorphisms. We first show that the variables learned by CRL methods cannot in general be used directly as input to common equation discovery methods, e.g., SiNDy (Brunton et al., 2016), which typically assume that the variables will lead to sparse equations. To tackle this problem, we introduce SPEED-AE (SParse

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

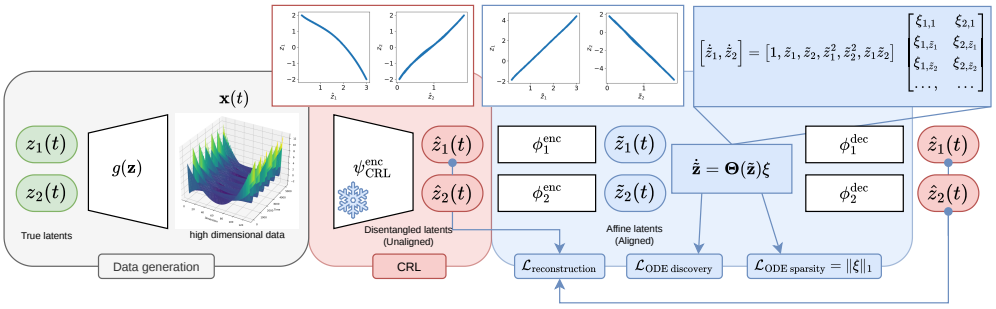


Figure 1: SPEED-AE pipeline: we observe high-dimensional data from an invertible mapping $\mathbf{x}(t) = g(\mathbf{z}(t))$ of the true latents $\mathbf{z}(t)$. First, SPEED-AE employs a pretrained CRL model to obtain low-dimensional latents $\hat{\mathbf{z}}(t)$ (red block). We assume that the recovered latents correspond to the ground-truth $\mathbf{z}$ up to a permutation and component-wise diffeomorphism $z_i = \hat{h}_i(\hat{z}_{\pi(i)})$. For simplicity, in the figure we assume that the permutation is the identity function. SPEED-AE learns a set of component-wise autoencoders $(\phi_i^{\text{enc}}, \phi_i^{\text{dec}})$ and an ODE by combining the functions in a library $\Theta(\tilde{\mathbf{z}})$, imposing sparsity of this representation (blue block). The output latents $\tilde{\mathbf{z}}(t)$ identify the ground-truth up to permutation and monomial and, in most cases, linear transformation.

Equivalent Equation Discovery AutoEncoder) a framework that combines a pretrained CRL method with a component-wise autoencoder that learns transformations of variables that are amenable to sparse ODE discovery. Figure 1 shows our SPEED-AE works: the frozen CRL model, which identifies the latents up to permutation and component-wise diffeomorphism, is followed by a component-wise AE. In the resulting latent space, the ODE is built from a predefined library, with sparsity imposed to recover an equation close to the true one.

We also investigate the identifiability of these learned variables and their equations, showing that, in general, we cannot have any guarantees on their relation to the ground truth ones, even if we assume sparsity in the equations or disentangled representations. In Lemma 3.1 we show that for polynomial ODEs, considering sparsity in the learned equations allows us to restrict the identifiability of each variable from component-wise polynomial to monomial diffeomorphisms. Experiments on Lotka-Volterra, Lorenz, and a two-pendulum system show that SPEED-AE improves on the disentanglement of the CRL methods and that it recovers ODEs that are closest to the ground truth, while achieving state-of-the-art forecasting performance.

2 Background

Ordinary Differential Equations (ODEs). Dynamical systems are commonly represented by ODEs (Strogatz, 2019), which model the evolution of their state $\mathbf{z}$ through an equation of the form

$$\dot{\mathbf{z}}(t) = \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} = \boldsymbol{f}(\mathbf{z}(t)), \qquad \mathbf{z}(0) = \mathbf{z}_0, \tag{1}$$

where $\mathbf{z}_0$ is the initial state. We focus on *autonomous* ODEs, for which $\boldsymbol{f}$ does not explicitly depend on t, so that $\dot{\mathbf{z}}(t)$ solely depends on the current state $\mathbf{z}(t)$. If $\boldsymbol{f}$ is Lipschitz continuous with respect to $\mathbf{z}$, the ODE admits a unique solution. Here, we address the recovery of the underlying ODE of a low-dimensional system $\mathbf{z}(t) \in \mathcal{Z} \subseteq \mathbb{R}^d$ having access only to high-dimensional trajectories of unstructured data $\mathbf{x}(t) = g(\mathbf{z}(t))$ with $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ and $D \gg d$, where $g : \mathcal{Z} \to \mathcal{X}$ is an unknown invertible mixing function, as assumed in Causal Representation Learning (Schölkopf et al., 2021).

Equation Discovery. ODE discovery aims to recover the symbolic expression of a dynamical system from measurements of the state variables $\mathbf{z}$ along one or more trajectories. SINDy (Brunton et al., 2016) applies a library Θ of L candidate functions, usually polynomials and some trigonometric functions, to the state variables $\mathbf{z}$ and seeks a sparse coefficient matrix Ξ by minimizing

$$\mathcal{L}_{\text{SINDy}}(\Xi) = \left\|\dot{\mathbf{z}} - \Xi^\top \Theta(\mathbf{z})\right\|_2^2 + \beta_1 \left\|\Xi\right\|_0 . \tag{2}$$

Here, $\Theta(\mathbf{z}) \in \mathbb{R}^L$ contains the candidate functions evaluated at $\mathbf{z}$, while $\Xi \in \mathbb{R}^{L \times d}$ selects them through its coefficients; the L_0 term promotes sparsity. In practice, derivatives $\dot{\mathbf{z}}$ are unavailable and must be estimated from data, e.g. via finite differences. SINDyAE extends SINDy to recover ODEs on hidden state variables $\mathbf{z} \in \mathbb{R}^d$ from high-dimensional observations $\mathbf{x} \in \mathbb{R}^D$ (Champion et al., 2019), jointly learning an encoder $\psi^{\text{enc}} \colon \mathbb{R}^D \to \mathbb{R}^d$ and a decoder $\psi^{\text{dec}} \colon \mathbb{R}^d \to \mathbb{R}^D$ by minimizing

$$\mathcal{L}_{\text{SINDyAE}}(\Xi, \psi^{\text{enc}}, \psi^{\text{dec}}) = \underbrace{\left\|\mathbf{x} - \psi^{\text{dec}}(\psi^{\text{enc}}(\mathbf{x}))\right\|_2^2}_{\text{Reconstruction loss}} + \beta_{\mathbf{z}} \underbrace{\left\|\left[\nabla \psi^{\text{enc}}(\mathbf{x})\right] \dot{\mathbf{x}} - \Xi^\top \Theta(\psi^{\text{enc}}(\mathbf{x}))\right\|_2^2}_{\text{SINDy loss in } \dot{\mathbf{z}}}$$
$$+ \beta_{\mathbf{x}} \underbrace{\left\|\dot{\mathbf{x}} - \left[\nabla \psi^{\text{dec}}(\psi^{\text{enc}}(\mathbf{x}))\right] \Xi^\top \Theta(\psi^{\text{enc}}(\mathbf{x}))\right\|_2^2}_{\text{SINDy loss in } \dot{\mathbf{x}}} + \beta_1 \underbrace{\|\Xi\|_1}_{\text{sparsity}}. \tag{3}$$

Here $[\nabla \psi^{\text{enc}}(\mathbf{x})] \in \mathbb{R}^{d \times D}$ and $[\nabla \psi^{\text{dec}}(\psi^{\text{enc}}(\mathbf{x}))] \in \mathbb{R}^{D \times d}$ are the encoder and decoder Jacobians evaluated at $\mathbf{x}$ and $\psi^{\text{enc}}(\mathbf{x})$, respectively; $\{\beta_{\mathbf{x}}, \beta_{\mathbf{z}}, \beta_1\}$ weigh the different loss terms. The L_1 penalty, coupled with sequential thresholding, is used as a smooth proxy of the L_0 optimization. Pervez et al. (2024) and Chen et al. (2024) propose a related approach that replaces derivative fitting by a Mechanistic Neural Network (MNN) that learns the coefficients Ξ and efficiently simulates trajectories. In our experiments, we compare to SINDyAE and MNNAE, which we further discuss in Appendix C.

Causal Representation Learning (CRL). CRL methods (Schölkopf et al., 2021) can provably recover the ground truth latent variables $\mathbf{z} \in \mathbb{R}^d$ from observations $\mathbf{x} \in \mathbb{R}^D$ up to certain specific classes of transformations, under different sets of assumptions or types of data. A CRL method typically learns a disentangled encoder $\psi^{\text{enc}} \colon \mathbb{R}^D \to \mathbb{R}^d$ where the latent space $\hat{\mathbf{z}} = \psi^{\text{enc}}(\mathbf{x})$ relates to the ground-truth variables $\mathbf{z}$ through a diffeomorphism $h \colon \mathbb{R}^d \to \mathbb{R}^d$. We focus on methods guaranteeing *identifiability up to permutation and component-wise diffeomorphisms*, i.e., $z_i = h_i(\hat{z}_{\pi(i)})$ for each $i \in \{1, \dots, d\}$ and permutation π. This type of identifiability guarantee on the learned variables is often also called *disentanglement*. In our experiments, we consider methods that use temporal data and interventions or actions to identify these variables, in particular CITRIS (Lippe et al., 2022) and DMSVAE (Lachapelle et al., 2022; 2026), which we also discuss in App. D, but in general our method is agnostic to the specific CRL method with the same identifiability class.

3 Identifiability Analysis for Polynomial ODEs

We study the identifiability of systems governed by a latent ODE $\dot{\mathbf{z}} = \boldsymbol{f}(\mathbf{z})$ from high-dimensional observations $\mathbf{x}(t) = g(\mathbf{z}(t))$, where g is an invertible mixing function. Given a smooth invertible encoder $\tilde{\mathbf{z}} = \psi^{\text{enc}}(\mathbf{x})$, the map h from the latent space $\tilde{\mathbf{z}}$ to the ground-truth variables $\mathbf{z}$ is a diffeomorphism. Consequently, as we derive in Prop. A.1, the encoded latent variables $\tilde{\mathbf{z}}$ are also governed by an ODE $\dot{\tilde{\mathbf{z}}} = \tilde{\boldsymbol{f}}(\tilde{\mathbf{z}}) = [\nabla h(\tilde{\mathbf{z}})]^{-1} \boldsymbol{f}(h(\tilde{\mathbf{z}}))$. Formally, whenever the variables of the two ODEs are related by a diffeomorphism, we say that the two ODEs are *conjugate* (Sideris, 2013).

By jointly learning an autoencoder $\psi^{\text{dec}} \circ \psi^{\text{enc}}$ and an ODE $\tilde{\boldsymbol{f}}$ over the latent space $\tilde{\mathbf{z}} = \psi^{\text{enc}}(\mathbf{z})$, SINDyAE (Champion et al., 2019) effectively recovers a diffeomorphically conjugate ODE. However, despite enforcing sparsity in $\tilde{\boldsymbol{f}}$, SINDyAE does not provide any guarantee on the recovery of the form of the true ODE or the ground-truth variables. As we show in a toy example in App. A.2, even for polynomial ODEs, the SINDyAE loss cannot guarantee that the learned variables correspond one-to-one to the state variables. Intuitively, this suggests that the equations learned on these variables might in general be of a more complicated form than the ground truth ones.

CRL methods learn a disentangled representation $\hat{\mathbf{z}}$ that identifies the true latents $\mathbf{z}$ up to permutation π and component-wise diffeomorphism $z_i = h_i(\hat{z}_{\pi(i)})$. Yet, even if we used an equation discovery method, e.g., SINDy (Brunton et al., 2016), directly on these variables and the learned ODE $\hat{\boldsymbol{f}}$, this could potentially still have an arbitrary form. In particular, given the transformations $\{h_i\}_{i=1}^d$ induced by a perfectly disentangled encoder $\psi^{\text{enc}}_{\text{CRL}}$, we can always write a conjugate ODE as

$$\dot{\hat{z}}_{\pi(i)} = \hat{f}_{\pi(i)}(\hat{\mathbf{z}}) = [h_i'(\hat{z}_{\pi(i)})]^{-1} f_i(h_1(\hat{z}_{\pi(1)}), \dots, h_d(\hat{z}_{\pi(d)})) \quad \text{for } i = 1, \dots, d, \tag{4}$$

where $h_i'(\hat{z}_{\pi(i)})$ are the the component-wise derivatives. Thus, even with perfect disentanglement, we cannot in general guarantee that the learned ODE will be similar to the ground truth one. We

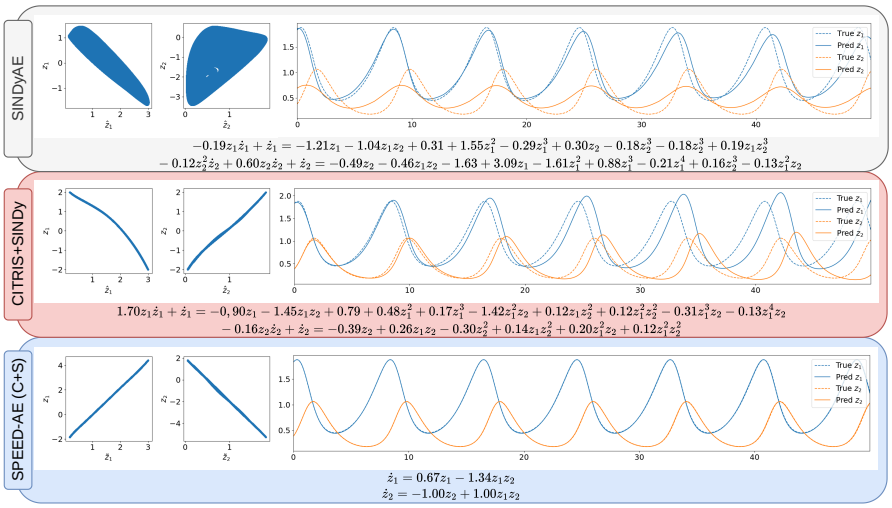


Figure 2: Comparison between SINDyAE, SINDy applied to CRL latents (using CITRIS), and SPEED-AE (with CITRIS as CRL method and SINDy as equation discovery method) on the Lotka-Volterra dataset ($\dot{z_1} = 0.66z_1 - 1.33z_1z_2$, $\dot{z_2} = -z_2 + z_1z_2$). For each model, the left box reports scatter plots of the true versus learned latents. We then compare a test trajectory with the one predicted from only the initial condition and the discovered equation, after converting both to the true latent space, following the evaluation protocol in Section 6. SPEED-AE is the only method that recovers a sparse equation and achieves strong long-term prediction performance.

therefore restrict ourselves to polynomial ODEs and assume that these component-wise diffeomorphisms are polynomial. In particular, we show that if the learned representations are disentangled and governed by a sparse polynomial ODE, we can identify the true latents up to component-wise monomial transformations. For tractability reasons, we introduce two assumptions (see Ass. A.3): after the change of variables, powers of polynomials and their full expansion do not undergo cancellations of terms (as per our remarks in App. A.3, this happens rarely).

Lemma 3.1. *Let $\dot{\mathbf{z}} = \boldsymbol{f}(\mathbf{z})$ and $\dot{\tilde{\mathbf{z}}} = \tilde{\boldsymbol{f}}(\tilde{\mathbf{z}})$ be two polynomial ODEs such that for every $i \in [d]$ $z_i = h_i(\tilde{z}_{\pi(i)})$ for an invertible polynomial h_i and permutation π. Let $q_i(z_i)$ be the highest degree polynomial such that $f_i(\mathbf{z}) = q_i(z_i)r_i(\mathbf{z})$, where $r_i(\mathbf{z})$ is the remainder polynomial and $N(r_i)$ be the number of monomial terms in $r_i(\mathbf{z})$. Similarly, we define $\tilde{r}_{\pi(i)}(\tilde{\mathbf{z}})$ and $N(\tilde{r}_{\pi(i)})$ for $\tilde{f}_{\pi(i)}$. Under Ass. A.2 and A.3, $N(\tilde{r}_{\pi(i)}) \geq N(r_i)$. If $N(\tilde{r}_{\pi(i)}) = N(r_i)$, then h_i is a monomial transformation, i.e., $z_i = h_i(\tilde{z}_{\pi(i)}) = a_i\tilde{z}_{\pi(i)}^{p_i}$ with $p_i \in \mathbb{N}^+$. Moreover, if q_i is constant, then $p_i = 1$, and h_i is affine.*

The proof is in App. A.3. Intuitively, Lemma 3.1 states that for polynomial ODEs, if the learned variables $\tilde{\mathbf{z}}$ correspond to the ground truth ones $\mathbf{z}$ up to permutations and component-wise polynomial diffeomorphisms, and we learn $\tilde{\boldsymbol{f}}(\tilde{\mathbf{z}})$ that is as sparse as possible (i.e., the number of its active monomial terms is as low as possible, which is $N(\tilde{r}_{\pi(i)}) = N(r_i)$), then this will lead to component-wise transformations that are monomials. Although this result holds only for polynomial ODEs and polynomial diffeomorphisms, we conjecture that if a representation is informed by sparsity and the form of the governing ODE, it results in a simpler component-wise transformation than with a standard causal representation learning method. The intuition of learning transformations of variables that have sparse ODE equations is similar to SINDyAE (Champion et al., 2019), but we can now provide theoretical guarantees on the identifiability of the variables up to simple indeterminacies.

4 Sparse Equivalent Equation Discovery AutoEncoder

The analysis in Sec. 3 shows that learning sparse equations does not provide any guarantees on the learned variables, while disentanglement of the learned variables in general does not provide guarantees on the learned equations. Moreover, as shown in Fig. 2, the representations learned by CRL methods might not be amenable to equation discovery methods, leading to complicated equations

that are far from the ground truth, even in simple cases, like polynomial ODEs. So we propose SParse Equivalent Equation Discovery AutoEncoder (SPEED-AE), which introduces an additional step after a pretrained CRL method: a component-wise autoencoder that learns a transformation of each CRL variable, such that these transformed variables have sparse ODE equations.

As a first step, SPEED-AE runs a CRL method to obtain latents $\hat{\mathbf{z}} = \psi_{\text{CRL}}^{\text{enc}}(\mathbf{x})$ that we assume recover the true variables up to permutation and component-wise diffeomorphism $z_i = \hat{h}_i(\hat{z}_{\pi(i)})$. Using these variables, it learns component-wise autoencoders $\tilde{z}_i = \phi_i^{\text{enc}}(\hat{z}_i)$ together with a sparse ODE over $\tilde{z}_i$, via an equation discovery model. The component-wise autoencoders of SPEED-AE guarantee that the new latents $\tilde{\mathbf{z}}$ are still disentangled with respect to the true variables $\mathbf{z}$. Our model does not assume any latent ODE form. However, under the assumptions of Lemma 3.1, SPEED-AE identifies the true latents up to simple monomial transformations. SPEED-AE is agnostic about the CRL methods (as long as they guarantee identifiability up to permutation and component-wise diffeomorphisms) and about the equation discovery approaches.

$$\mathcal{L}_{\text{SPEED-AE}} = \sum_{i=1}^{d} \left\| \phi^{\text{dec}}\left(\phi^{\text{enc}}(\hat{z}_i)\right) - \hat{z}_i \right\|_2^2, + \mathcal{L}_{\text{ODE discovery}} + \mathcal{L}_{\text{ODE sparsity}}. \tag{5}$$

For instance, we can use a SINDy-based loss representing the ODE as $\dot{\mathbf{z}} = \Xi^\top \Theta(\tilde{\mathbf{z}})$, where $\Theta(\tilde{\mathbf{z}}) \in \mathbb{R}^L$ is a library of L functions of $\tilde{\mathbf{z}}$ weighted by coefficients $\Xi \in \mathbb{R}^{L \times d}$:

$$\mathcal{L}_{\text{ODE discovery}} = \beta_{\mathbf{z}} \left\| [\nabla \phi^{\text{enc}}(\hat{\mathbf{z}})](\dot{\hat{\mathbf{z}}}) - \Xi^\top \Theta(\tilde{\mathbf{z}}) \right\|_2^2 + \beta_{\mathbf{x}} \left\| \dot{\hat{\mathbf{z}}} - [\nabla \phi^{\text{dec}}(\tilde{\mathbf{z}})](\Xi^\top \Theta(\tilde{\mathbf{z}})) \right\|_2^2. \tag{6}$$

The same framework can instead use an MNN-based loss:

$$\mathcal{L}_{\text{ODE discovery}} = \beta_{\mathbf{z}} \left\| \tilde{\mathbf{z}} - \text{MNN}(\tilde{\mathbf{z}}(0)) \right\|_2^2 + \beta_{\mathbf{x}} \left\| \hat{\mathbf{z}} - \phi^{\text{dec}}(\text{MNN}(\phi^{\text{enc}}(\hat{\mathbf{z}}(0))) \right\|_2^2, \tag{7}$$

where $\text{MNN}(\tilde{\mathbf{z}}_0)$ is the solution to the learned ODE $\dot{\tilde{\mathbf{z}}} = \Xi^\top \Theta(\tilde{\mathbf{z}})$ simulated by the MNN. In both variants, the second term, weighted by $\beta_{\mathbf{x}}$, prevents collapse of $\tilde{\mathbf{z}}$ by enforcing consistency between dynamics in $\tilde{\mathbf{z}}$ and their push-forward $\hat{\mathbf{z}}$, similarly to Champion et al. (2019). For the sparsity loss, we follow Brunton et al. (2016); Champion et al. (2019); Chen et al. (2024) and use an L_1 penalty with sequential thresholding as a proxy for the L_0 norm. The penalty sparsifies Ξ, while coefficients below the threshold are periodically set to 0. To reflect the setting of Sec. 3, we first implement the encoder ϕ^{enc} and decoder ϕ^{dec} as learnable polynomial functions. However, our results on simple synthetic experiments (App. G) are unsatisfactory, as these representations were not flexible enough during training. Hence, in our final model, we use MLPs for both ϕ^{enc} and ϕ^{dec}. This is also justified empirically that the CRL methods we use only guarantee identifiability up to general diffeomorphism only. Thus, it is not enough to only consider polynomial transformations and the autoencoder has to represent a more general function. Implementation details are in App. B.

5 Related Work

Equation discovery. Learning ODE systems from data has been extensively studied when we can observe the state variables directly, both from a theoretical (Scholl et al., 2024) and a practical point of view (Brunton et al., 2016; Kaheman et al., 2020; Chen et al., 2024). SINDY (Brunton et al., 2016), the principal sparse regression approach, considers a library of functions evaluated on the variables $\Theta(\hat{\mathbf{z}})$ (usually polynomials and some trigonometric functions) and learns the coefficients Ξ of a linear model that predicts the variables' derivatives $\dot{\hat{\mathbf{z}}}$ from the linear combination of the functions in the library $\Xi^\top \Theta(\hat{\mathbf{z}})$. With the implicit assumption that most dynamical systems in nature have a sparse representation, a sparsity regularization on ξ is employed to select the simplest model among all solutions. Mechanistic Neural Networks (MNNs, (Pervez et al., 2024; Chen et al., 2024)) are general approaches to model the evolution of dynamical systems by building an internal ODE representation. The mechanistic encoder maps the trajectory into the coefficients and other parameters of the ODE, whose solution is then calculated by solving an equivalent linear system efficiently. By using the state variables $\mathbf{z}$ directly as the internal ODE variables and matching the available trajectories, the MNN can be effectively used for equation discovery (Pervez et al., 2024) based on a library of basis functions. Finally, symbolic recovery approaches (Becker et al., 2023; D'Ascoli et al., 2022; 2024) treat variables and operation operators (such as addition, multiplication, exponential) as tokens in a transformer-based regression approach. Recent work on theoretical results

for ODE identifiability focuses on linear or affine ODEs (Qiu et al., 2022; Duan et al., 2020), and on particular settings such as unobserved variables (Wang et al., 2024a), discrete observations (Wang et al., 2024c), or sparse coefficients Casolo et al. (2025). By contrast, theoretical identifiability in non-linear ODEs remains limited to known functional forms (Grewal & Glover, 1976; Stanhope et al., 2014) or single initial conditions (Yao et al., 2024).

Equation discovery on unstructured data. Few works address settings where variables are not directly observed, and only high-dimensional measurements (e.g., images or videos) are available. Existing approaches either learn equations on latent variables without guarantees that these variables correspond to the ground-truth (Champion et al., 2019; Auzina et al., 2023), or assume the underlying functional form of the ODE is known (Linial et al., 2021) and focus on parameter identification. The most related work to us, SindyAE (Champion et al., 2019) uses an autoencoder to learn representations of the state variables in the latent space, on which it then learns ODEs with Sparse Identification for Nonlinear Dynamics (SINDy) (Brunton et al., 2016). Similarly to us it encourages learning representations that can be used in sparse equations, but as opposed to us it does not provide any theoretical guarantee or analysis on the quality of the learned representations.

Causal Representation Learning. Causal Representation Learning (CRL) (Schölkopf et al., 2021) methods provably identify a latent representation $\hat{\mathbf{z}}$ that identifies the true latents $\mathbf{z}$ up to permutation and a component-wise transformation. CRL methods exploit multi-view and multi-environment settings (Kügelgen et al., 2021; Xu et al., 2024; Yao et al., 2023), interventional data (Lippe et al., 2022; von Kügelgen et al., 2023; Ahuja et al., 2023; Squires et al., 2023), temporal data and actions (Lippe et al., 2023; Lachapelle et al., 2022; 2026) or partially observable settings (Xu et al., 2024). While CRL methods focus on identifiability of state variables from unstructured data, they generally do not learn ODE systems, and do not leverage sparsity in the resulting equations to improve identifiability, as we do. Recently, Yao et al. (2024) identify time-invariant trajectory-specific parameters, but then assume that the state variables are directly observed and that the trajectories are generated from a single initial condition. In our work we do not assume neither of these assumptions, but we then assume that the parameters of the ODE we are learning are the same across all trajectories. These works are therefore complementary and could be potentially combined.

6 Experiments

We evaluate SPEED-AE on different versions of three dynamical systems: Lotka-Volterra, Lorenz, and two pendulum dynamical systems, but instead of observing the state directly, we observe high-dimensional measurements $\mathbf{x}$, produced by an invertible mixing function on the true state variables $\mathbf{z}$. We discuss in detail the data generation and model training in App. E, including the pretraining of the CRL models in each setting by considering a separate dataset of trajectories that also contains interventions on each variable z_i with probability $p = 0.01$.

We consider two CRL approaches, CITRIS (Lippe et al., 2022) and DMSVAE (Lachapelle et al., 2022; 2026) and two ODE discovery approaches, SINDy (Brunton et al., 2016) and MNN (Pervez et al., 2024; Chen et al., 2024). We denote each combination as SPEED-AE(X+Y), where X indicates the CRL method (C for CITRIS or D for DMSVAE), and Y indicates the equation discovery method (S for SINDy or M for MNN). We compare with the only two baselines that learn both the latents $\hat{\mathbf{z}}$ and the governing equations without additional information: SINDyAE Champion et al. (2019) and MNNAE, which combines the MNN with an AE, discussed in App. C. In the original SINDyAE, the true image derivatives $\dot{\mathbf{x}}$ are provided as input; we instead use empirical derivatives to better reflect realistic conditions. Moreover, we also compare, as an ablation, a direct combination of the CRL methods that we use with the equation discovery methods that we use.

We evaluate three different metrics: (a) correlation discrepancy between the ground truth and learned variables, (b) trajectory prediction, and (c) ODE recovery. We define the correlation discrepancy ($CorrD$) as the MAE between the Pearson correlation of the ground truth variables $\mathbf{z}$ and the cross-correlation matrix between the ground truth $\mathbf{z}$ and learned variables $\tilde{\mathbf{z}}$, defined as $CorrD = \frac{1}{d^2}\sum_{i,j}^{d} |\rho(\mathbf{z}_i, \tilde{\mathbf{z}}_j) - \rho(\mathbf{z}_i, \mathbf{z}_j)|$. We use this metric to evaluate the quality of disentanglement of the state variables, while allowing for dependences between each ground truth state variables. Intuitively, lower is better and $CorrD = 0$ corresponds to perfect disentangelment. For trajectory prediction and ODE recovery, we have to first address a key challenge: after training, each

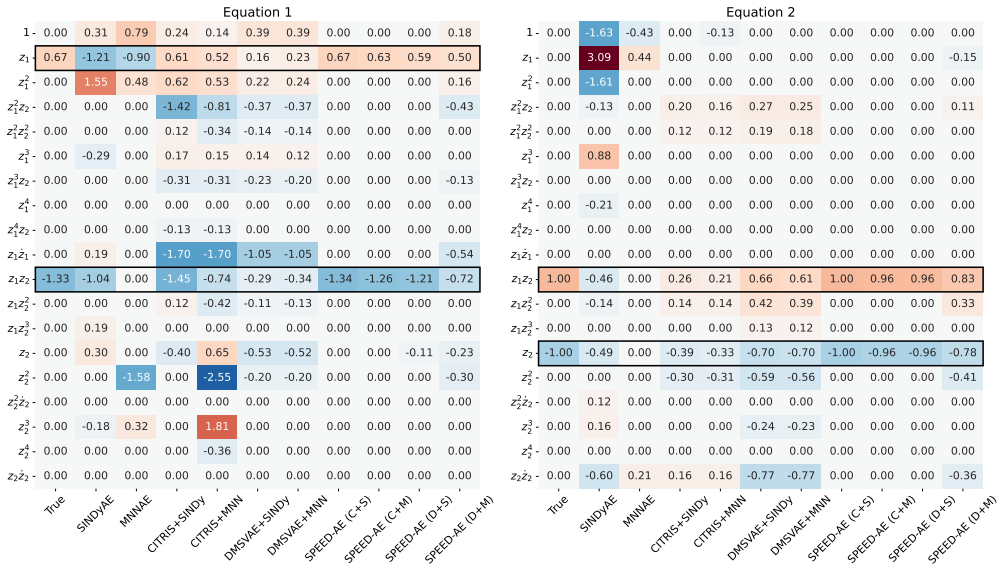


Figure 3: Learned coefficients for Lotka-Volterra for each polynomial of the state variables (each row) and different methods (each column, where the last four are versions of SPEED-AE). The first column shows the ground truth coefficients and the black boxes highlight the non-zero coefficients.

model represents the system in its own latent space, which means that they are not directly comparable. To compare them, we therefore convert all of the learned representations to the same space, the space of true latents $\mathbf{z}$. Here we summarize our evaluation and provide more details in App. F.

For trajectory prediction evaluation, we first learn a component-wise MLP σ_i mapping each learned variable $\tilde{\mathbf{z}}_i$ to the ground-truth variable $\mathbf{z}_{\pi(i)}$ by choosing a permutation π that maximises their correlation. We then compute the latent prediction error $\mathbf{z}$Err as the L^2 distance between the true trajectories $\mathbf{z}(t)$ and the predicted trajectories projected to the same space $\sigma(\tilde{\mathbf{z}})(t)$, averaging over all timesteps t. We then compare the observation prediction error $\mathbf{x}$Err, as the L^2 distance between the observations $\mathbf{x}(t)$ and predicted observations $\tilde{\mathbf{x}}(t) = \psi^{\text{dec}}(\phi^{\text{dec}}(\tilde{\mathbf{z}}))(t)$, where ϕ^{dec} is the SPEED-AE decoder and ψ^{dec} is the CRL decoder.

For ODE discovery, we first learn d component-wise polynomial maps $\tilde{z}_i = \text{Poly}_i(z_i)$, then express the recovered ODEs $\dot{\tilde{\mathbf{z}}} = \tilde{\boldsymbol{f}}(\tilde{\mathbf{z}})$ via a change of variable through the learned polynomial, as shown in Prop. A.1. We compare the resulting coefficients with those of the true ODE using the Sum of Absolute and Squared Errors (CoeffSAE and CoeffSSE). Since both the ODEs and the maps are polynomial, the converted ODE remains polynomial, although possibly implicit. In this case, the error increases due to the presence of terms such as $z_i\dot{z}_i$, which are penalised. The best case is when the map is affine, $\tilde{z}_i = a_i z_i + c_i$, which always leads to an explicit ODE and lower coefficient error. Since SINDyAE and MNNAE do not define an assignment between $\tilde{\mathbf{z}}$ and $\mathbf{z}$, we select the permutation with the best Mean Correlation Coefficient before learning the map.

App. G reports ablations on the CRL latents and SPEED-AE architecture. With perfect disentanglement, SPEED-AE achieves negligible errors in CorrD and ODE recovery. This indicates that performance from our models is only limited by the disentanglement of the CRL method. Our experiments with polynomial encoders and decoders show that, even with perfect disentanglement and the simplest diffeomorphisms between true and CRL latents, these maps are not flexible or stable enough during training to provide good results, motivating the choice of $\phi^{\text{enc}}, \phi^{\text{dec}}$ being MLPs.

6.1 Lotka-Volterra and Lorenz with polynomial libraries

We consider two systems with polynomial libraries of order 3: Lotka-Volterra and Lorenz. The Lotka-Volterra model (Strogatz, 2019) is a system of two first-order ODEs

$$\dot{z}_1 = \alpha z_1 - \beta z_1 z_2, \qquad \dot{z}_2 = -\gamma z_2 + \delta z_1 z_2, \tag{8}$$

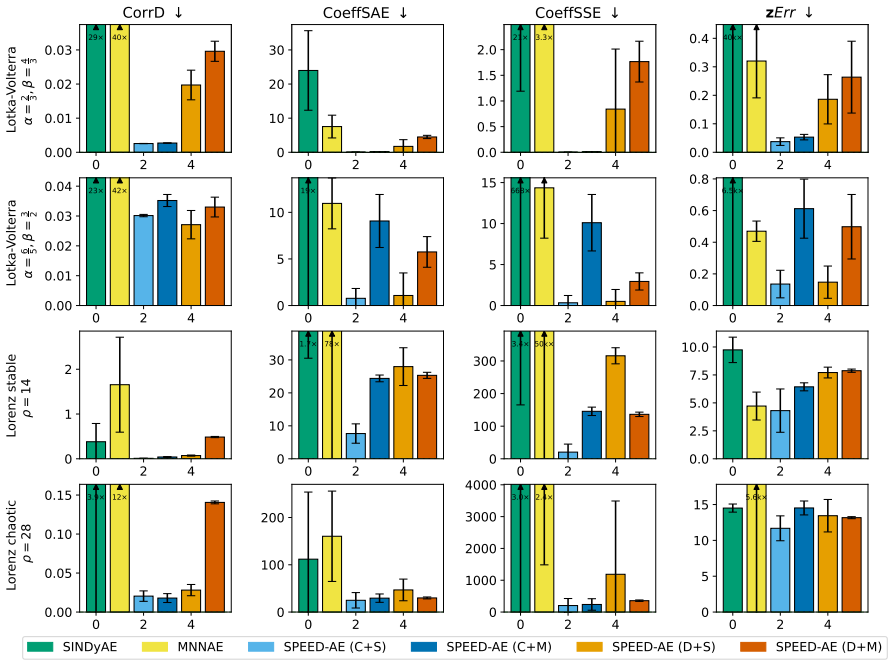


Figure 4: Results for the Lotka-Volterra and Lorenz experiments with mean and standard deviation over 10 seeds. For all metrics, lower is better. We limit the height of the boxes for better readability, reporting the actual scales of the outlier results inside their bars.

for prey z_1 and predator z_2 populations. We fix $\delta = \gamma = 1$ and consider two cases for α, β. In the first, we set $\alpha = \frac{2}{3}, \beta = \frac{4}{3}$ and train the whole pipeline. In the second, we set $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$ but use the same CRL models from the first case, i.e., we train only the component-wise autoencoders and the equation discovery components of SPEED-AE, while the baselines are trained from scratch. The objective is to show that SPEED-AE allows us to transfer learned disentangled representations to new dynamics without performance drops. We generate 120 trajectories of 5000 steps with $\Delta t = 0.01$ and an $80/20/20$ split. The observation $\mathbf{x}$ is generated as in the Lorenz setting of Champion et al. (2019). On $[-1, 1]$, we evaluate the first four Legendre polynomials on a grid of 128 points, obtaining $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{u}_4 \in \mathbb{R}^{128}$, and define

$$\mathbf{x}(t) = \mathbf{u}_1 z_1(t) + \mathbf{u}_2 z_2(t) + \mathbf{u}_3 z_1^3(t) + \mathbf{u}_4 z_2^3(t) \in \mathbb{R}^{128}. \tag{9}$$

This map from $\mathbb{R}^2$ to its image in $\mathbb{R}^{128}$ is non-linear, smooth, and invertible on its image, since Legendre polynomials are a complete basis in $[-1, 1]$ and are therefore linearly independent. The Lorenz system (Strogatz, 2019) is a system of three ODEs describing atmospheric convection:

$$\dot{z}_1 = \sigma(z_2 - z_1), \quad \dot{z}_2 = z_1(\rho - z_3) - z_2, \quad \dot{z}_3 = z_1 z_2 - \beta z_3, \tag{10}$$

where σ, ρ, β are physical fluid constants. We set $\sigma = 10, \beta = \frac{8}{3}$, and consider $\rho = 14$ and $\rho = 28$, which produce stable and chaotic trajectories, respectively. Following Champion et al. (2019), we generate trajectories of 250 time steps with $\Delta t = 0.02$. The observations are generated in the same way as Champion et al. (2019) and analogous to the Lotka-Volterra setup. With 3 variables, we use 6 Legendre polynomials, obtaining a non-linear, smooth and map between $\mathbb{R}^3$ and $\mathbb{R}^{128}$.

Fig. 3 shows the learned coefficients for both equations of Lotka-Volterra for all of the evaluated methods. SPEED-AE recovers ODEs that after conversion, are closest to the ground truth, especially when using CITRIS, while the baselines recover more complicated equations with multiple unnecessary terms. In particular, the coefficients of the CRL methods used directly with the equation discovery methods (e.g., CITRIS + SINDy) perform worse than the corresponding versions of our framework (e.g., SPEED-AE (C + S)), showing the effectiveness of SPEED-AE and that disentanglement alone is insufficient The results for Lorenz are in App. H.2, showing similar trends.

Fig. 4 shows that for Lotka-Volterra all SPEED-AE variants beat the baselines by a large margin, except for MNNAE on zErr. However, MNNAE's strong performance on forecasting comes at the

Table 1: Results for the Pendulum experiment for the model.

Model	CorrD	**x**Err	**z**Err	Recovered ODE
SINDyAE	0.431	18.888	1.621	$\ddot{z}_1 - 0.302\ddot{z}_1 z_1 + 1.20\ddot{z}_1 z_1^2 + 2.39\ddot{z}_1 \dot{z}_1 z_1 =$ $= -0.32 + 0.75z_2 - 0.43z_1 + 0.17z_1^3 + 0.06z_1^2 + \dots$ $\ddot{z}_2 + 0.01\ddot{z}_2 z_2 + 0.03\ddot{z}_2 z_2^2 + 0.05\ddot{z}_2 \dot{z}_2 z_2 =$ $= 0.03 + 0.18z_1 - 0.07z_1^3 - 0.03z_1^2 + \dots$
CITRIS+SINDy	0.020	6.824	0.239	$\ddot{z}_1 - 0.15\dot{z}_1^2 + 0.15\ddot{z}_1 z_1 + 0.02\dot{z}_1^2 z_1^2 - 0.012\dot{z}_2^2 z_2^2 + \dots =$ $= 0.64 + 11.31z_1 - 131z_1^2 + \cos(0.14\dot{z}_1 z_1 - 0.93z_1) + \dots$ $\ddot{z}_2 - 0.16\ddot{z}_2 z_2 - 0.16\dot{z}_2^2 + 0.32\dot{z}_2 z_2 + \dots =$ $= 1.48 + 4.03z_2 - 0.04z_2^2 + \cos(0.14z_2\dot{z}_2 + 0.84z_2 - 0.43) \dots$
SPEED-AE (C+S)	0.004	6.155	0.177	$\ddot{z}_1 = -0.27z_1 - 0.61\sin(1.05z_1)$ $\ddot{z}_2 = -0.27z_2 - 0.61\sin(1.07z_2)$

cost of no identifiable latents. For Lorenz, the **stable** case ($\rho = 14$), correlation and ODE recovery results (Figure 4) show that SPEED-AE outperforms both baselines. The slight performance drop of DMSVAE-based variants reflects a weaker disentanglement compared to CITRIS (see App. H.2). The strong forecasting results of MNNAE are again countered by poor correlation and ODE recovery. Under **chaotic** dynamics ($\rho = 28$), SPEED-AE remains unmatched in ODE recovery. Forecasting results are similar to the stable case, except that MNNAE fails in more than one seed.

We report the results on the **x**Err metric in App. H, where we also compare with a Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) and a Transformer (Vaswani et al., 2017), which learn to predict $\mathbf{x}$ directly. These models are effective on short trajectories, such as the Lorenz ones, but collapse on the longer ones from Lotka-Volterra, where recovering the underlying set of variables and ODEs yields stable long-horizon performance with errors up to 20 times lower.

6.2 Pendulums and non-polynomial libraries

We test SPEED-AE on a second-order system and actual image data. We consider a system of two independent pendulums governed by $\ddot{z}_1 = -\sin(z_1)$ and $\ddot{z}_2 = -\sin(z_2)$, similarly to Champion et al. (2019). Images have resolution 64×64 and separate channels for the two pendulums. We use a library that includes $\sin(z)$, as in SINDyAE, the main baseline for this experiment. Given their consistently strong performance, we only use CITRIS for CRL and SINDy for ODE discovery.

Tab. 1 shows that SPEED-AE reduces forecasting errors by a factor of $3 - 4$ relative to SINDyAE, which fails to identify and disentangle the two pendulums, instead learning only one variable correlated with a pendulum (see App. H.3). The ODEs recovered by SPEED-AE are nearly identical to each other and to the the true ones, reflecting that the pendulums share the same dynamics. Although CITRIS+SINDy forecasts accurately, its equations are less simple and interpretable than those of SPEED-AE, providing empirical support for our discussion in Sec. 3 beyond the polynomial case.

7 Conclusion

We show that neither disentanglement nor sparsity alone is enough to simultaneously learn the true structural variables and their governing equations from high-dimensional data. We propose SPEED-AE, a pipeline that learns a componentwise transformation of disentangled variables from a any CRL model that is informed by sparsity of the learned equations given a library of functions. SPEED-AE achieves the best performance among baselines, especially in terms of correlation with the true latents and the recovered ODEs. While our methodology is general, as any CRL and ODE recovery methods can be plugged into our pipeline, one of our limitations it that the results depend on the performance of each step. In particular, a poor disentanglement can hinder the final performance, although SPEED-AE can drastically improve CRL results even in this case. Moreover, our theoretical analysis is limited by the non-cancellation assumptions and polynomial diffeomorphisms. We believe the empirical and theoretical results of this work represent a step forward towards learning meaningful and transferable representations of dynamical systems from unstructured data and that these results can be further improved in future works by the community.

AI USE STATEMENT

In this work, we used generative AI tools for: (i) finding counterexamples or possible trivial extensions to the main results; (ii) polishing and correcting writing and grammar; (iii) helping with the implementation of the ML baselines, the modularization of the code, finding bugs, and some visualization scripts.

We have **not** used generative AI tools for: (i) the idealization, conceptualization, and implementation of the core methodology and framework; (ii) the theoretical discussion, the main theoretical claims and their proofs; (iii) the development of the evaluation pipeline, which constitutes another key contribution; (iv) data generation; (v) support in the data analysis and interpretation of results.

We have reviewed all AI-assisted work. For example, we checked the counterexamples ourselves and manually checked and tested the generated code. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work aims to advance the field of Machine Learning for Science and Dynamical Systems. The data, methodology, and results in this paper do not pose safety risks and do not raise questions or concerns regarding the Code of Ethics.

REPRODUCIBILITY STATEMENT

We provide the complete experimental details in Section 6, Appendix E, and Appendix F, which provide all the information needed to implement the method and reproduce the experiments, including data generation. We include the details on the hardware in Appendix E.1. Finally, we provide all the code and the scripts to run the experiments as Supplementary Material.

REFERENCES

John Abbott. Sparse Squares of Polynomials. *Mathematics of Computation*, 71(237):407–413, 2002. ISSN 0025-5718.

Kartik Ahuja, Divyat Mahajan, Yixin Wang, and Yoshua Bengio. Interventional Causal Representation Learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 372–407. PMLR, July 2023.

Ilze Amanda Auzina, Cagatay Yildiz, Sara Magliacane, Matthias Bethge, and Efstratios Gavves. Modulated Neural ODEs. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Sören Becker, Michal Klein, Alexander Neitz, Giambattista Parascandolo, and Niki Kilbertus. Predicting Ordinary Differential Equations with Transformers. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 1978–2002. PMLR, July 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, April 2016. doi: 10.1073/pnas.1517384113.

Cecilia Casolo, Sören Becker, and Niki Kilbertus. Identifiability Challenges in Sparse Linear Ordinary Differential Equations. In *The Fourteenth International Conference on Learning Representations*, October 2025.

Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116 (45):22445–22451, November 2019. doi: 10.1073/pnas.1906995116.

Jiale Chen, Dingling Yao, Adeel Pervez, Dan Alistarh, and Francesco Locatello. Scalable Mechanistic Neural Networks. In *The Thirteenth International Conference on Learning Representations*, October 2024.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Don Coppersmith and James Davenport. Polynomials whose powers are sparse. *Acta Arithmetica*, 58(1):79–87, 1991. ISSN 0065-1036.

Stéphane D'Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, and Francois Charton. Deep symbolic regression for recurrence prediction. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 4520–4536. PMLR, June 2022.

Stéphane D'Ascoli, Sören Becker, Philippe Schwaller, Alexander Mathis, and Niki Kilbertus. ODEFormer: Symbolic Regression of Dynamical Systems with Transformers. *International Conference on Learning Representations*, 2024:21943–21976, May 2024.

Brian De Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton. PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. *Journal of Open Source Software*, 5(49):2104, May 2020. ISSN 2475-9066. doi: 10.21105/joss.02104.

X Duan, J E Rubin, and D Swigon. Identification of affine dynamical systems from a single trajectory. *Inverse Problems*, 36(8):085004, August 2020. ISSN 0266-5611. doi: 10.1088/1361-6420/ab958e.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented Neural ODEs. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

M. Grewal and K. Glover. Identifiability of linear and nonlinear dynamical systems. *IEEE Transactions on Automatic Control*, 21(6):833–837, December 1976. ISSN 1558-2523. doi: 10.1109/TAC.1976.1101375.

Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ODE models with Gaussian processes. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1959–1968. PMLR, July 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

Kadierdan Kaheman, J. Nathan Kutz, and Steven L. Brunton. SINDy-PI: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2242):20200279, October 2020. ISSN 1364-5021. doi: 10.1098/rspa.2020.0279.

Alan Kaptanoglu, Brian De Silva, Urban Fasel, Kadierdan Kaheman, Andy Goldschmidt, Jared Callaham, Charles Delahunt, Zachary Nicolaou, Kathleen Champion, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton. PySINDy: A comprehensive Python package for robust sparse system identification. *Journal of Open Source Software*, 7(69):3994, January 2022. ISSN 2475-9066. doi: 10.21105/joss.03994.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017.

Julius Von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-Supervised Learning with Data Augmentations Provably Isolates Content from Style. In *Advances in Neural Information Processing Systems*, November 2021.

Sebastien Lachapelle, Pau Rodriguez, Yash Sharma, Katie E. Everett, Rémi LE Priol, Alexandre Lacoste, and Simon Lacoste-Julien. Disentanglement via Mechanism Sparsity Regularization: A New Principle for Nonlinear ICA. In *Proceedings of the First Conference on Causal Learning and Reasoning*, pp. 428–484. PMLR, June 2022.

Sébastien Lachapelle, Pau Rodríguez López, Yash Sharma, Katie Everett, Rémi Le Priol, Alexandre Lacoste, and Simon Lacoste-Julien. Nonparametric Partial Disentanglement via Mechanism Sparsity: Sparse Actions, Interventions and Sparse Temporal Dependencies. *Journal of Machine Learning Research*, 27(71):1–90, 2026. ISSN 1533-7928.

Ori Linial, Neta Ravid, Danny Eytan, and Uri Shalit. Generative ODE Modeling with Known Unknowns. In *Proceedings of the Conference on Health, Inference, and Learning*, pp. 79–94, April 2021. doi: 10.1145/3450439.3451866.

Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M. Asano, Taco Cohen, and Stratis Gavves. CITRIS: Causal Identifiability from Temporal Intervened Sequences. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 13557–13603. PMLR, June 2022.

Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M. Asano, Taco Cohen, and Efstratios Gavves. BISCUIT: Causal Representation Learning from Binary Interactions. In *The 39th Conference on Uncertainty in Artificial Intelligence*, June 2023.

Adeel Pervez, Francesco Locatello, and Stratis Gavves. Mechanistic Neural Networks for Scientific Machine Learning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 40484–40501. PMLR, July 2024.

Xing Qiu, Tao Xu, Babak Soltanalizadeh, and Hulin Wu. Identifiability analysis of linear ordinary differential equation systems with a single trajectory. *Applied Mathematics and Computation*, 430:127260, October 2022. ISSN 0096-3003. doi: 10.1016/j.amc.2022.127260.

Alfréd Rényi. On the Minimal Number of Terms of the Square of a Polynomial. *Hungarica Acta Mathematica*, 1:30–34, 1947. ISSN 0001-5954.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Andrzej Schinzel and Umberto Zannier. On the number of terms of a power of a polynomial. *Rendiconti Lincei*, 20(1):95–98, March 2009. ISSN 1120-6330. doi: 10.4171/rlm/534.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward Causal Representation Learning. *Proceedings of the IEEE*, 109(5):612–634, May 2021. ISSN 1558-2256. doi: 10.1109/JPROC.2021.3058954.

Philipp Scholl, Aras Bacho, Holger Boche, and Gitta Kutyniok. Symbolic Recovery of Differential Equations: The Identifiability Problem, October 2024.

Thomas C. Sideris. *Ordinary Differential Equations and Dynamical Systems*, volume 2 of *Atlantis Studies in Differential Equations*. Atlantis Press, Paris, 2013. ISBN 978-94-6239-020-1 978-94-6239-021-8. doi: 10.2991/978-94-6239-021-8.

Chandler Squires, Anna Seigal, Salil S. Bhate, and Caroline Uhler. Linear Causal Disentanglement via Interventions. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 32540–32560. PMLR, July 2023.

S. Stanhope, J. E. Rubin, and D. Swigon. Identifiability of Linear and Linear-in-Parameters Dynamical Systems from a Single Trajectory. *SIAM Journal on Applied Dynamical Systems*, 13(4): 1792–1815, January 2014. doi: 10.1137/130937913.

Steven Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. A Chapman & Hall Book. CRC Press, Boca Raton London New York, second edition, first issued in hardback edition, 2019. ISBN 978-0-8133-4910-7 978-0-367-09206-1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. Van Der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul Van Mulbregt, SciPy 1.0 Contributors, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius De Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, and Yoshiki Vázquez-Baeza. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020. ISSN 1548-7091, 1548-7105. doi: 10.1038/s41592-019-0686-2.

Julius von Kügelgen, Michel Besserve, Wendong Liang, Luigi Gresele, Armin Kekić, Elias Bareinboim, David Blei, and Bernhard Schölkopf. Nonparametric Identifiability of Causal Representations from Unknown Interventions. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Yuanyuan Wang, Biwei Huang, Wei Huang, Xi Geng, and Mingming Gong. Identifiability Analysis of Linear ODE Systems with Hidden Confounders. *Advances in Neural Information Processing Systems*, 37:59054–59092, December 2024a.

Yuanyuan Wang, Wei Huang, Mingming Gong, Xi Geng, Tongliang Liu, Kun Zhang, and Dacheng Tao. Identifiability and Asymptotics in Learning Homogeneous Linear ODE Systems from Discrete Observations. *Journal of Machine Learning Research*, 25(154):1–50, 2024b. ISSN 1533-7928.

Yuanyuan Wang, Wei Huang, Mingming Gong, Xi Geng, Tongliang Liu, Kun Zhang, and Dacheng Tao. Identifiability and Asymptotics in Learning Homogeneous Linear ODE Systems from Discrete Observations, June 2024c.

Danru Xu, Dingling Yao, Sebastien Lachapelle, Perouz Taslakian, Julius Von Kügelgen, Francesco Locatello, and Sara Magliacane. A Sparsity Principle for Partially Observable Causal Representation Learning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 55389–55433. PMLR, July 2024.

Dingling Yao, Danru Xu, Sebastien Lachapelle, Sara Magliacane, Perouz Taslakian, Georg Martius, Julius von Kügelgen, and Francesco Locatello. Multi-View Causal Representation Learning with Partial Observability. In *The Twelfth International Conference on Learning Representations*, October 2023.

Dingling Yao, Caroline Muller, and Francesco Locatello. Marrying Causal Representation Learning with Dynamical Systems for Science. *Advances in Neural Information Processing Systems*, 37: 71705–71736, December 2024.

Umberto Zannier. On composite lacunary polynomials and the proof of a conjecture of Schinzel. *Inventiones mathematicae*, 174(1):127–138, October 2008. ISSN 1432-1297. doi: 10.1007/s00222-008-0136-8.

A PROOFS OF THE THEORETICAL STATEMENTS AND DISCUSSION

In this Section, we provide proofs of the theoretical statements and discuss the limits and challenges of the problems in Section 3.

A.1 PRELIMINARIES

First, we report a simple proposition that shows that for any ODE $\dot{\mathbf{z}} = \boldsymbol{f}(\mathbf{z})$ defined on $\mathbf{z}$ and any diffeomorphism h that maps $\tilde{\mathbf{z}}$ to $\mathbf{z}$, we can define a conjugated ODE with the change of variable formula. Intuitively this means that learning an ODE on our learned variables $\tilde{\mathbf{z}}$ does not provide any constraints in terms of the class of functions h.

Proposition A.1 (Change of Variable)**.** *Consider an ODE $\dot{\mathbf{z}} = \boldsymbol{f}(\mathbf{z})$ defined on $\mathbf{z}$, and let h be a diffeomorphism that maps a different set of variables $\tilde{\mathbf{z}}$ to $\mathbf{z} = h(\tilde{\mathbf{z}})$. Then, given a solution $\mathbf{z}(t)$ with initial condition $\mathbf{z}_0$, the transformed trajectory $\tilde{\mathbf{z}} = h^{-1}(\mathbf{z})$ solves the following Cauchy problem:*

$$\dot{\tilde{\mathbf{z}}} = [\nabla h(\tilde{\mathbf{z}})]^{-1} \boldsymbol{f}(h(\tilde{\mathbf{z}})) \qquad \hat{\mathbf{z}}(0) = h^{-1}(\mathbf{z}_0) \tag{11}$$

Proof. It is sufficient to consider the time derivative of $\mathbf{z} = h(\tilde{\mathbf{z}})$ and apply the chain rule on the right-hand side, obtaining

$$\dot{\mathbf{z}} = [\nabla h(\tilde{\mathbf{z}})]\dot{\tilde{\mathbf{z}}}. \tag{12}$$

Then, we can substitute the ODE for $\mathbf{z}$ on the left-hand side, obtaining

$$\boldsymbol{f}(\mathbf{z}) = [\nabla h(\tilde{\mathbf{z}})]\dot{\tilde{\mathbf{z}}}. \tag{13}$$

Finally, we substitute $\mathbf{z} = h(\tilde{\mathbf{z}})$:

$$[\nabla h(\tilde{\mathbf{z}})]\dot{\tilde{\mathbf{z}}} = \boldsymbol{f}(h(\tilde{\mathbf{z}})). \tag{14}$$

Since h is a diffeomorphism, $[\nabla h(\tilde{\mathbf{z}})]$ is invertible everywhere and thus $\dot{\tilde{\mathbf{z}}} = [\nabla h(\tilde{\mathbf{z}})]^{-1} \boldsymbol{f}(h(\tilde{\mathbf{z}}))$. □

A.2 SINDYAE LOSS CANNOT GUARANTEE DISENTANGLEMENT FOR POLYNOMIAL ODES

We show that following two polynomial ODEs are equivalent in terms of the SINDyAE loss, since they are related by a diffeomorphism and have the same number of terms

$$\boldsymbol{f}(\mathbf{z}) = \begin{pmatrix} -z_2 + z_1^3 + z_1 z_2^2 \\ z_1 + z_1^2 z_2 + z_2^3 \end{pmatrix} \qquad \tilde{\boldsymbol{f}}(\tilde{\mathbf{z}}) = \begin{pmatrix} \tilde{z}_2 + 2\tilde{z}_1^3 + 2\tilde{z}_1\tilde{z}_2^2 \\ -\tilde{z}_1 + 2\tilde{z}_1^2\tilde{z}_2 + 2\tilde{z}_2^3, \end{pmatrix} \tag{15}$$

although their representations are not disentangled (as we see below).

Let $A = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, consider the diffeomorphism

$$\mathbf{z} = h(\tilde{\mathbf{z}}) \;=\; A\tilde{\mathbf{z}} \;=\; \big(\tilde{z}_1 + \tilde{z}_2,\; \tilde{z}_1 - \tilde{z}_2\big).$$

Since h is a linear transformation, it holds $\nabla h(\tilde{\mathbf{z}}) = A$ for every $\tilde{\mathbf{z}}$. Furthermore, since $A^{-1} = \frac{1}{2}A$, we can also compute $[\nabla h(\tilde{z})]^{-1} = \frac{1}{2}A$. We rewrite slightly the original polynomial ODE as

$$\boldsymbol{f}(\mathbf{z}) = \begin{pmatrix} -z_2 + z_1^3 + z_1 z_2^2 \\ z_1 + z_1^2 z_2 + z_2^3 \end{pmatrix} = J\mathbf{z} + |\mathbf{z}|^2\mathbf{z},$$

using matrix $J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ and notation $|\mathbf{z}|^2 = z_1^2 + z_2^2$.

We can write the conjugate ODE, using the change of variable form from Proposition A.1, as

$$\tilde{\boldsymbol{f}}(\tilde{\mathbf{z}}) = [\nabla h(\tilde{z})]^{-1} \boldsymbol{f}(h(\tilde{z})) \tag{16}$$
$$= \frac{1}{2} A \left(JA\tilde{\mathbf{z}} + |A\tilde{\mathbf{z}}|^2 A\tilde{\mathbf{z}} \right) \tag{17}$$
$$= \frac{1}{2} AJA\tilde{\mathbf{z}} + \frac{1}{2} A|A\tilde{\mathbf{z}}|^2 A\tilde{\mathbf{z}} \tag{18}$$
$$= -J\tilde{\mathbf{z}} + \frac{1}{2} A|A\tilde{\mathbf{z}}|^2 A\tilde{\mathbf{z}} \tag{19}$$
$$= -J\tilde{\mathbf{z}} + \frac{1}{2} A2|\tilde{\mathbf{z}}|^2 A\tilde{\mathbf{z}} \tag{20}$$
$$= -J\tilde{\mathbf{z}} + |\tilde{\mathbf{z}}|^2 AA\tilde{\mathbf{z}} \tag{21}$$
$$= -J\tilde{\mathbf{z}} + |\tilde{\mathbf{z}}|^2 2I\tilde{\mathbf{z}} \tag{22}$$
$$= -J\tilde{\mathbf{z}} + 2|\tilde{\mathbf{z}}|^2 \tilde{\mathbf{z}} \tag{23}$$
$$= \begin{pmatrix} \tilde{z}_2 + 2\tilde{z}_1^3 + 2\tilde{z}_1\tilde{z}_2^2 \\ -\tilde{z}_1 + 2\tilde{z}_1^2\tilde{z}_2 + 2\tilde{z}_2^3. \end{pmatrix} \tag{24}$$

In particular, even if both ODE systems are polynomial and each component of h is a polynomial function, the SINDy loss cannot guarantee on its own that the learned variables $\tilde{\mathbf{z}}$ are disentangled, i.e., they correspond to the ground truth variables $\mathbf{z}$ up to permutation π and component-wise transformations h_i that only depend on a single $z_{\pi(i)}$. This suggests that the learned equations on these variables might in general be more complicated than the ground truth ones.

A.3 Identifiability of Latent variables and ODE

Before proceeding to the proof of the main result, we pose the following assumptions on the polynomial functions involved in our problem.

Assumption A.2 (No cancellations in Powers of Polynomials)**.** Let $h_i(\tilde{z}_{\pi(i)}) = \sum_{k \leq p} \beta_k \tilde{z}_{\pi(i)}^k$ be any of the component-wise polynomial diffeomorphisms that we consider, with degree p. Then, the symbolic representation of $h_i(\tilde{z}_{\pi(i)})^\ell$ has the maximum possible number of terms, i.e., no terms cancel out.

Consider the square of a generic polynomial of degree $d = 2$: $(1 + \beta_1 z + \beta_2 z^2)^2$, which expands to

$$1 + 2\beta_1 z + (\beta_1^2 + 2\beta_2) z^2 + 2\beta_1\beta_2 z^3 + \beta_2^2 z^4.$$

Then, under assumption A.2, all the monomials appearing in the expansion must have a non-zero coefficient. In other terms, it implies that $\beta_1 \neq 0$, $\beta_2 \neq 0$, and $\beta_1^2 + 2\beta_2 \neq 0$. This assumption does not say that in every possible polynomial power, every possible term z^k with degree between 0 and $p\ell$ has a non-zero coefficient. For example, $(1 + \beta_2 z^2 + \beta_4 z^4)^2$ expands to a polynomial containing only only even powers of z. Here, Assumption A.2 indicates that all even exponents appear. We remark that this assumption is not restrictive since, if we assume that the coefficients of $P(z)$ are chosen at random in some interval, having cancellations has zero probability. Finally, we remark that powers of polynomials are an active research topic in algebra (Zannier, 2008; Schinzel & Zannier, 2009). Without assumptions, no bounds on the number of terms in a power of a polynomial would be helpful for our theoretical analysis. In particular, by accurately selecting the coefficients of $P(z)$, one can even find cases where $P(z)^\ell$ has fewer terms than $P(z)$ (Rényi, 1947; Coppersmith & Davenport, 1991; Abbott, 2002).

Assumption A.3 (Variable changes do not annihilate terms)**.** Consider component-wise polynomial maps $z_i = h(\tilde{z}_{\pi(i)})$. In the context of Lemma 3.1, with $f_i(z_i) = q_i(z_i) r_i(\mathbf{z})$, when performing the variable change on the system of ODEs, obtaining a new system of equations

$$\dot{\tilde{z}}_{\pi(i)} = \frac{q_i(h_i(\tilde{z}_{\pi(i)}))}{h_i'(\tilde{z}_{\pi(i)})} r_i(h_1(\tilde{z}_{\pi(1)}), \ldots, h_d(\tilde{z}_{\pi(d)})), \tag{25}$$

all monomial terms that appear in the symbolic expansion of $r_i(h(\tilde{\mathbf{z}}_{\pi(i)}))$ have non-zero coefficients.

Similar remarks apply to Assumption A.3, where cancellation of terms in the expansion of $r_i(h(\mathbf{z}))$ requires specific conditions on the coefficients of r_i and h_j, which have measure zero when the coefficients are sampled randomly. For instance, the ODE $f(z) = z + z^2 - z^3$ can be decomposed as $f(z) = q(z)r(z)$ where $q(z) = z$ and $r(z) = 1 + z - z^2$. The diffeomorphism $h(\tilde{z}) = \tilde{z} + \alpha$ produces the conjugate ODE $\dot{\tilde{z}} = \left(\alpha + \alpha^2 - \alpha^3\right) + \left(1 + 2\alpha - 3\alpha^2\right)\tilde{z} + (1 - 3\alpha)\,\tilde{z}^2 - \tilde{z}^3$, which in turn factorizes as $[h'(\tilde{z})]^{-1} q(h(\tilde{z}))h'(\tilde{z})r(h(\tilde{z}))$, where

$$q(h(\tilde{z})) = \tilde{z} + \alpha \tag{26}$$

$$h'(\tilde{z}) = 1 \tag{27}$$

$$r(h(\tilde{z})) = \left(1 + \alpha - \alpha^2\right) + (1 - 2\alpha)\,\tilde{z} - \tilde{z}^2. \tag{28}$$

Hence, satisfying Assumption A.3 requires that $1 + \alpha - \alpha^2 \neq 0$ and $1 - 2\alpha \neq 0$.

Then, before proving our main result, we introduce a technical lemma, which sets a lower bound on the number of terms in the symbolic representation of a power of a polynomial.

Lemma A.4 (Number of terms in a polynomial power)**.** *Let $P(z)$ be a polynomial with degree p, $P(z) = \sum_{j \leq p} \beta_j z^j$ and let n be the number of non-zero coefficients. Then, assuming that no cancellations happen (Assumption A.2), the number of terms in $P(z)^\ell$ is at least $1 + (n-1)\ell$.*

Proof. We assume that the polynomial $P(z)$ has a non-zero constant term β_0. If not, it is sufficient to factor out the smallest power of z and work with the remaining polynomial, which has the same number of terms. Therefore, we prove the statement by induction on the power ℓ:

- **Base case.** $\ell = 1$. The statement holds as the polynomial has $1 + (n-1)\ell = n$ terms.
- **Inductive step** $\ell \implies \ell + 1$. Consider the $\ell + 1$-th power as the following product:

$$P(z)^{\ell+1} = P(z)^\ell P(z) = \left(\sum_{j \leq \ell p} \gamma_j z^j\right)\left(\beta_0 + \dots \beta_p z^p\right), \tag{29}$$

where the first parentheses is the result of expanding $P(z)^\ell$. We now expand this product by considering two particular terms: one given by the highest-degree term of $\beta_p z^p$ of $P(z)$ multiplied by $P(z)^\ell$ and the other given by the remainder of the difference $P(z) - \beta_p z^p$ multiplied by γ_0:

$$\begin{aligned} P(z)^{\ell+1} &= \left(\sum_{j \leq \ell p} \gamma_j z^j\right)\left(\beta_0 + \dots \beta_p z^p\right) \\ &= \beta_p z^p \left(\sum_{j \leq \ell p} \gamma_j z^j\right) + \gamma_0\left(\beta_0 + \dots \beta_{p-1} z^{p-1}\right) + Q(z), \end{aligned} \tag{30}$$

where $Q(z)$ is the remainder of terms. The first term of this sum involves only terms with degree at least p and has the same number of terms as $P(z)^\ell$, which, by inductive hypothesis, is at least $1 + (n-1)\ell$. The second term, instead, has one fewer term than $P(z)$ and maximum degree $p - 1$. Assuming no cancellations (Assumption A.2), the number of terms in equation 30 is at least the sum of the number of terms in its two terms. Hence, $P(z)^{\ell+1}$ has at least $1 + (n-1)\ell + (n-1) = 1 + (n-1)(l+1)$ terms as wanted.

□

In the following, we denote monomials through multi-indexes $\ell \in \mathbb{N}^d$ with degree $|\ell| = \sum_{i=1}^d \ell_i$. Therefore, we indicate multi-powers of vectors as $\mathbf{z}^\ell = z_1^{\ell_1} \cdots z_d^{\ell_d}$. We are now ready to prove our main result, which we restate here.

Lemma 3.1. *Let $\dot{\mathbf{z}} = \boldsymbol{f}(\mathbf{z})$ and $\dot{\tilde{\mathbf{z}}} = \tilde{\boldsymbol{f}}(\tilde{\mathbf{z}})$ be two polynomial ODEs such that for every $i \in [d]$ $z_i = h_i(\tilde{z}_{\pi(i)})$ for an invertible polynomial h_i and permutation π. Let $q_i(z_i)$ be the highest degree*

polynomial such that $f_i(\mathbf{z}) = q_i(z_i)r_i(\mathbf{z})$, where $r_i(\mathbf{z})$ is the remainder polynomial and $N(r_i)$ be the number of monomial terms in $r_i(\mathbf{z})$. Similarly, we define $\tilde{r}_{\pi(i)}(\tilde{\mathbf{z}})$ and $N(\tilde{r}_{\pi(i)})$ for $\tilde{f}_{\pi(i)}$. Under Ass. A.2 and A.3, $N(\tilde{r}_{\pi(i)}) \geq N(r_i)$. If $N(\tilde{r}_{\pi(i)}) = N(r_i)$, then h_i is a monomial transformation, i.e., $z_i = h_i(\tilde{z}_{\pi(i)}) = a_i \tilde{z}_{\pi(i)}^{p_i}$ with $p_i \in \mathbb{N}^+$. Moreover, if q_i is constant, then $p_i = 1$, and h_i is affine.

Proof. We start from the implicit version of the transformed ODE (Prop. A.1) for a generic index i:

$$[h_i'(\tilde{z}_{\pi(i)})]\dot{\tilde{z}}_{\pi(i)} = f_i(h_1(\tilde{z}_{\pi(1)}), \dots, h_d(\tilde{z}_{\pi(d)})). \tag{31}$$

For the new ODE to be polynomial, it is necessary that

$$\dot{\tilde{z}}_{\pi(i)} = \tilde{f}_i(\tilde{\mathbf{z}}) = \frac{f_i(h_1(\tilde{z}_{\pi(1)}), \dots, h_d(\tilde{z}_{\pi(d)}))}{h_i'(\tilde{z}_{\pi(i)})} \tag{32}$$

is a polynomial and, thus, that $f_i(h_1(\tilde{z}_{\pi(1)}), \dots, h_d(\tilde{z}_{\pi(d)}))$ is divisible by $h_i'(\tilde{z}_{\pi(i)})$. Most importantly, if $h_i'(\zeta) = 0$ for some $\zeta \in \mathbb{R}$, i.e., ζ is a root of $h_i'(\tilde{z}_{\pi(i)})$, we must have that

$$f_i(h_1(\tilde{z}_{\pi(1)}), \dots, h_i(\zeta), \dots, h_d(\tilde{z}_{\pi(d)})) = 0 \tag{33}$$

for every value of the other $\tilde{z}_j$, as otherwise $\tilde{f}_i$ would not be a polynomial. Thus, $h_i(\zeta)$ always annihilates $f_i(\mathbf{z})$, and $z_i - h_i(\zeta)$ must divide f_i.

We now consider the decomposition $f_i(\mathbf{z}) = q_i(z_i)r_i(\mathbf{z})$ where $q_i(z_i)$ has the highest degree. In this composition, it must be that $z_i - h_i(\zeta)$ does not divide $r_i(\mathbf{z})$, as otherwise we could write $f_i(\mathbf{z}) = q_i(z_i)(z_i - h_i(\zeta))r_i'(\mathbf{z})$ and q_i would not have the highest degree possible. Thus, we must have $h_i'(\tilde{z}_{\pi(i)}) | q_i(h_i(\tilde{z}_{\pi(i)}))$ and

$$\tilde{f}(\tilde{\mathbf{z}}) = \frac{q_i(h_i(\tilde{z}_{\pi(i)}))}{h_i'(\tilde{z}_{\pi(i)})} r_i(h(\tilde{\mathbf{z}}_\pi)) \tag{34}$$

and, for $\tilde{f}$ to be a polynomial, $h_i'(\tilde{z}_{\pi(i)})$ must divide $q_i(h_i(\tilde{z}_{\pi(i)}))$, leaving $r_i(h(\tilde{\mathbf{z}}_\pi))$ untouched. Thus, we can decompose the conjugated ODE as $\tilde{f}(\tilde{\mathbf{z}}) = \tilde{q}_i(\tilde{z}_{\pi(i)})\tilde{r}_i(\tilde{\mathbf{z}})$, where

$$\tilde{q}_i(\tilde{z}_{\pi(i)}) = \frac{q_i(h_i(\tilde{z}_{\pi(i)}))}{h_i'(\tilde{z}_{\pi(i)})} \tag{35}$$

$$\tilde{r}_i(\tilde{\mathbf{z}}) = r_i(h(\tilde{\mathbf{z}}_\pi)). \tag{36}$$

Thus, the sparsity in $\tilde{f}_{\pi(i)}$ is strongly related to the sparsity of $\tilde{r}_{\pi(i)}(\tilde{\mathbf{z}}) = r_i(h(\tilde{\mathbf{z}}_\pi))$. In general, $r_i(\mathbf{z})$ has the form $r_i(\mathbf{z}) = \sum_{|\ell| \leq \deg(f_i)} \alpha_{i,\ell} z_1^{\ell_1} \cdots z_d^{\ell_d}$. We start by expanding $r_i(h_1(\tilde{z}_{\pi(1)}), \dots, h_d(\tilde{z}_{\pi(d)}))$, where we recall $h_j(\tilde{z})$ are all polynomials as well:

$$\begin{aligned} \tilde{r}_i(\tilde{\mathbf{z}}) = r_i(h_1(\tilde{z}_{\pi(1)}), \dots, h_d(\tilde{z}_{\pi(d)})) &= \sum_{|\ell| \leq \deg(\boldsymbol{f})} \alpha_{i,\ell}\, h_1(\tilde{z}_{\pi(1)})^{\ell_1} \cdots h_d(\tilde{z}_{\pi(d)})^{\ell_d} \\ &= \sum_{|\ell| \leq \deg(\boldsymbol{f})} \alpha_{i,\ell} \prod_{j=1}^{d} \left(\sum_{k \leq p_j} \beta_{jk} \tilde{z}_{\pi(j)}^k \right)^{\ell_j}. \end{aligned} \tag{37}$$

We now focus on a single term related to the coefficient $\alpha_{i,\ell}$. Our objective is to calculate the total number of terms that will appear once we expand the calculations. To do so, we need to evaluate the number of terms resulting from taking the ℓ_j power of the polynomials. If we let n_j be the number of terms in $h_j(\tilde{z}_{\pi(j)})$, Lemma A.4 tells us that $h_j(\tilde{z}_{\pi(j)})^{\ell_j}$ has at least $1 + (n_j - 1)\ell_j$ terms. Furthermore, since they all involve a single variable and they are all different, the symbolic expansion of the product $h_1(\tilde{z}_{\pi(1)})^{\ell_1} \cdots h_d(\tilde{z}_{\pi(d)})^{\ell_d}$ will have the maximum number of terms possible, which is at least $\prod_{j=1}^{d}(1 + (n_j - 1)\ell_j)$. In the end, a single term $\alpha_{i,\ell}$ now accounts for at least $\prod_{j=1}^{d}(1 + (n_j - 1)\ell_j)$ new ones.

Additionally, we notice that the highest-degree monomial among these is given by $\prod_{j=1}^{d} \tilde{z}_{\pi(j)}^{p_j \ell_j}$. This is because this term comes from the product of the highest-degree monomial in each diffeomorphism

$h_j(\tilde{z}_{\pi(j)})$, which has degree p_j as defined above, and the exponent ℓ_j. We call this monomial term the *representative* of the multi-index ℓ.

Given two multi-indices $\ell^{(1)}, \ell^{(2)}$, their representative monomials are distinct (in terms of degrees of their variables). This is because if they were to be equal, we would have that, for each j, $\ell_j^{(1)} p_j = \ell_j^{(2)} p_j$ and, thus, $\ell_j^{(1)} = \ell_j^{(2)}$. In particular, since the representatives are all distinct, $\tilde{r}_{\pi(i)}$ has at least one term for each monomial in r_i, that is $N(\tilde{r}_{\pi(i)}) \geq N(r_i)$.

Clearly, we can have that different multi-indices $\ell^{(1)}, \ell^{(2)}, \dots, \ell^{(K)}$ all produce a monomial term related to some multi-index $\tilde{\ell}$ in the new variables $\tilde{\mathbf{z}}$. However, Assumption A.3 ensures that this term remains active and the contributions coming from the different $\ell^{(j)}$ do not cancel out. Hence, minimizing the number of terms in the transformed ODE requires minimizing $\prod_{k=1}^{d}(1+(n_j-1)l_j)$, as all the terms will survive.

Since the multi-indices ℓ, the corresponding powers ℓ_j, and the coefficients α_j are all given by the original ODE and are thus fixed, this leaves us with a single option for minimizing the number of terms after the change of variable: minimizing n_j, i.e., the number of terms in $h_j(\tilde{z}_{\pi(j)}) = \sum_{k=1}^{p_j} \beta_{jk} \tilde{z}_{\pi(j)}^k$. In fact, if $n_j = 1$, each power will involve exactly one term, and the ODE keeps the same number of terms. This means that, to have the minimal number of terms, we must have that all β_{jk} are zero except one, i.e., h_j are all monomial transformations $h_j(\tilde{z}_{\pi(j)}) = a_j z_{\pi(j)}^{p_j}$.

For the special case, we now look again at the structure of

$$\tilde{f}(\tilde{\mathbf{z}}) = \frac{q_i(h_i(\tilde{z}_{\pi(i)}))}{h_i'(\tilde{z}_{\pi(i)})} r_i(h(\tilde{\mathbf{z}}_\pi)) \tag{38}$$

and we ask when $h_i'(\tilde{z}_{\pi(i)})$ divides $q_i(h_i(\tilde{z}_{\pi(i)}))$. As stated earlier, whenever ζ is a root of $h_i(\tilde{z}_{\pi(i)})$, $h_i(\zeta)$ annihilates f_i and $z_i - h_i(\zeta)$ must divide f_i. This necessary condition limits which polynomial transformations h_i are admissible given the form of $f_i(z)$. In particular, if $q_i(z_i) = d_i$ is a constant, it means that $h_i'(\tilde{z}_{\pi(i)})$ must be a constant as well: $h_i'(\tilde{\mathbf{z}}) = a_i \in \mathbb{R}$. Thus, $h_i(\tilde{z}_{\pi(i)}) = a_i \tilde{z}_{\pi(i)} + b_i$, and h_i is necessarily an affine function. □

Counterexample to the general case. Lemma 3.1 shows that an affine map h_i can be recovered if $q_i(z_i)$ is a constant. One can also show that if q_i has a single root $d_i(z_i - c_i)^{k_i}$, then $h_i(\tilde{z}_i) = a_i(\tilde{z}_{\pi(i)} - b_i)^k + c_i$. This comes from the fact that, for $h_i'(\tilde{z}_{\pi(i)})$ to divide $q_i(h_i(\tilde{z}_{\pi(i)}))$, h' must have a single root, with arbitraty multiplicity.

One might wonder what happens if $q_i(z_i)$ is in a very general form. Unfortunately, we found no satisfactory property for $h_i(\tilde{z}_i)$ in this case, and we provide a counterexample in this sense.

Consider $\pi(i) = i$ and a system of polynomial ODEs with the i-th component being

$$\dot{z}_i = f_i(z) = (z_i + 2)(z_i - 2) r_i(\mathbf{z}). \tag{39}$$

For $h_i(\tilde{z}_i)$ to be admissible, the only condition is that $h'(\tilde{z}_i)$ must divide $(h_i(\tilde{z}_i) + 2)(h_i(\tilde{z}_i) - 2)$. For example, the polynomial $h(\tilde{z}_i) = \tilde{z}_i^3 - 3\tilde{z}_i$ fulfills these requests, and if restricted to an appropriate domain, constitutes a diffeomorphism. Even more complex cases can be constructed as, the divisibility condition provides a way to construct examples.

On the assumption that h is polynomial. In our theoretical results, we assume that h is polynomial, which might not be true, especially in the experimental settings. We remark that, in compact spaces, any function can be approximated up to arbitrary precision by a polynomial with sufficient degree.

We investigated whether this hypothesis is necessary for our theoretical analysis. In particular, we tried to understand if, given $\dot{z} = f(z)$ polynomial and $\dot{\tilde{z}} = \tilde{f}(\tilde{z})$ polynomial as well, it is necessary that $h(\tilde{z})$ is polynomial. Unfortunately, this is not the case, which makes the assumption necessary. Consider

$$\dot{z} = f(z) = 1, \tag{40}$$

Now, consider the mapping $z = h(\tilde{z}) = \arctan(\tilde{z})$. By applying the change of variable, we have

$$\frac{1}{1 + \tilde{z}^2} \dot{\tilde{z}} = 1 \tag{41}$$

and, consequently, $\dot{\tilde{z}} = 1 + \tilde{z}^2$. Both ODEs are polynomial, but the diffeomorphisms $h : \mathbb{R} \to (-\pi/2, \pi/2)$ and its inverse are not polynomial.

B SPEED-AE IMPLEMENTATION DETAILS

In this Section, we describe the implementation of SPEED-AE for the SINDy and MNN loss cases.

In each case, the architecture of SPEED-AE consists of a set of d component-wise encoders ϕ_i^{enc} and d component-wise decoders ϕ_i^{dec}, as well as the set of ODE coefficients $\boldsymbol{\Xi}$ for the library $\boldsymbol{\Theta}(\tilde{\mathbf{z}})$. The encoders map each variable $\hat{\mathbf{z}}_i$ coming from the disentangled CRL representation to a new encoding $\tilde{z}_i = \phi_i^{\text{enc}}(\hat{z}_i)$, while the decoders aim to recover the CRL variables $\hat{z}_i$. The ODE coefficients Ξ define the learned ODE as $\dot{\tilde{\mathbf{z}}} = \boldsymbol{\Theta}(\tilde{\mathbf{z}})^\top \Xi$, similar to (Brunton et al., 2016; Champion et al., 2019; Chen et al., 2024). Below, we describe the two specific modes we used to train SPEED-AE.

SINDy-based loss: SPEED-AE (X+S) We use a similar approach to Champion et al. (2019), where the key idea is to learn a SINDy-like loss $\left\| \dot{\tilde{\mathbf{z}}} - \boldsymbol{\Theta}(\tilde{\mathbf{z}})^\top \Xi \right\|_2$ both on the learned representations $\tilde{\mathbf{z}}$ and on the original variables $\hat{\mathbf{z}}$. To calculate this loss, we need to obtain the reference values for the derivatives $\dot{\tilde{\mathbf{z}}}$, which can be obtained via the chain rule $\dot{\tilde{\mathbf{z}}} = (\nabla_{\hat{\mathbf{z}}} \phi^{\text{enc}})(\dot{\hat{\mathbf{z}}})$. In contrast, the derivatives of the CRL-encoded trajectories $\dot{\hat{\mathbf{z}}}$ are calculated via finite differences with the pysindy library (De Silva et al., 2020; Kaptanoglu et al., 2022). The complete loss reads as:

$$\begin{aligned} \mathcal{L}_{\text{SPEED-AE(X+S)}} = &\sum_{i=1}^{d} \left\| \phi_i^{\text{dec}} \left(\phi_i^{\text{enc}}(\hat{z}_i) \right) - \hat{z}_i \right\|_2^2 + \beta_{\mathbf{z}} \left\| (\nabla_{\hat{\mathbf{z}}} \phi^{\text{enc}})(\dot{\hat{\mathbf{z}}}) - \Theta(\tilde{\mathbf{z}})^\top \Xi \right\|_2^2 + \\ &+ \beta_{\mathbf{x}} \left\| \dot{\hat{\mathbf{z}}} - (\nabla_{\tilde{\mathbf{z}}} \phi^{\text{dec}})(\Theta(\tilde{\mathbf{z}})^\top \Xi) \right\|_2^2 + \beta_1 \left\| \Xi \right\|_1 , \end{aligned} \tag{42}$$

where the names of the weights are chosen for consistency with those of MNNAE and SINDyAE. The first term imposes a low reconstruction error on the autoencoders, the second is the SINDy loss on $\dot{\tilde{\mathbf{z}}}$ with the push-forwarded derivatives from $\dot{\hat{\mathbf{z}}}$, and the third term calculates the SINDy loss on the original variables $\hat{\mathbf{z}}$, based on the push-forward of the SINDy derivatives through the decoder $\dot{\hat{\mathbf{z}}}_{\text{SINDy}} = (\nabla_{\tilde{\mathbf{z}}} \phi^{\text{dec}})(\Theta(\tilde{\mathbf{z}})^\top \Xi)$. The last term induces sparsity with the L^1 norm, while sequential thresholding is applied every 100 epochs.

MNN-based loss: SPEED-AE (X+M) In case of the MNN (Pervez et al., 2024; Chen et al., 2024), derivatives are not necessary for both $\tilde{\mathbf{z}}$ and $\hat{\mathbf{z}}$. Instead, the MNN calculates the trajectory from the initial condition and the ODE given by $\dot{\tilde{\mathbf{z}}} = \boldsymbol{\Theta}(\tilde{\mathbf{z}})^\top \Xi$. The complete loss reads at

$$\begin{aligned} \mathcal{L}_{\text{SPEED-AE(X+M)}} = &\sum_{i=1}^{d} \left\| \phi_i^{\text{dec}} \left(\phi_i^{\text{enc}}(\hat{z}_i) \right) - \hat{z}_i \right\|_2^2 + \beta_{\mathbf{z}} \left\| \tilde{\mathbf{z}} - \text{MNN}(\tilde{\mathbf{z}}(0)) \right\|_2^2 + \\ &+ \beta_{\mathbf{x}} \left\| \hat{\mathbf{z}} - \phi^{\text{dec}}(\text{MNN}(\phi^{\text{enc}}(\hat{\mathbf{z}}(0)))) \right\|_2^2 + \beta_1 \|\Xi\|_1, \end{aligned} \tag{43}$$

where the names of the weights are chosen for consistency with those of MNNAE and SINDyAE. Similar to the above case, the third term ensures that the AE and the MNN model work together to reconstruct the trajectories from the initial condition, avoiding the collapse of the learned representation. As in the SINDy-based loss, the last term induces sparsity with the L^1 norm, while sequential thresholding is applied every 100 epochs. For this model, the training dataset is generated by unfolding trajectories of a fixed length from the available trajectories, i.e., considering almost every point in the dataset as an initial condition.

C BASELINES

Here we provide the details on the definitions and implementation of the employed baselines.

SINDyAE. SINDyAutoencoder was introduced in Champion et al. (2019) to recover latent coordinates and governing equations in a single step. It consists of

- an Autoencoder, i.e., an encoder $\tilde{\mathbf{z}} = \psi_{\text{S}}^{\text{enc}}(\mathbf{z})$ and a decoder $\tilde{\mathbf{x}} = \psi_{\text{S}}^{\text{dec}}(\tilde{\mathbf{z}})$, both of which are implemented with MLPs.
- An underlying SINDy model, consisting of a library of functions $\Theta(\tilde{\mathbf{z}}) \in \mathbb{R}^{L \times d}$ and the learnable linear coefficients $\Xi \in \mathbb{R}^L$ that define the ODE as $\Theta(\tilde{\mathbf{z}})^\top \Xi$.

The model is trained with the following loss:

$$\begin{aligned}\mathcal{L}_{\text{SINDyAE}} = \left\|\mathbf{x} - \psi_{\text{S}}^{\text{enc}}\psi_{\text{S}}^{\text{dec}}(\mathbf{x})\right\|_2^2 + \beta_{\mathbf{x}} \left\|\dot{\mathbf{x}} - (\nabla_{\tilde{\mathbf{z}}}\psi_{\text{S}}^{\text{dec}})(\Theta(\tilde{\mathbf{z}})^\top \Xi)\right\|_2^2 + \\ + \beta_{\mathbf{z}} \left\|(\nabla_{\mathbf{x}}\psi_{\text{S}}^{\text{enc}})(\dot{\mathbf{x}}) - \Theta(\tilde{\mathbf{z}})^\top \Xi\right\|_2^2 + \beta_1 \left\|\Xi\right\|_1,\end{aligned} \tag{44}$$

where each term represents, in order:

- The reconstruction of the autoencoder.
- The loss on the derivative of $\mathbf{x}$, which is compared to the pushforward of the predicted derivative $\dot{\tilde{\mathbf{z}}} = \Theta(\tilde{\mathbf{z}})^\top \Xi$ through the decoder $\psi_{\text{S}}^{\text{dec}}$ with the chain rule.
- The loss on the derivative of $\mathbf{z}$, which compares the predicted derivative $\dot{\tilde{\mathbf{z}}} = \Theta(\tilde{\mathbf{z}})^\top \Xi$ with the pushforward of the ground truth one $\dot{\mathbf{x}}$ through the encoder $\psi_{\text{S}}^{\text{enc}}$.
- The sparsity loss on Ξ.

To actually sparsify the equations, every 500 epochs, the coefficients in Ξ below a certain threshold are zeroed and masked. We use the original implementation available on SINDyAutoencoder's GitHub.

MNNAE. We employ the original MNN model from Pervez et al. (2024); Chen et al. (2024) and combine it with an Autoencoder, i.e., similarly to the PDE solving and Discovery of Physical Parameters experiments in Pervez et al. (2024). MNNAE consists of:

- an Autoencoder, i.e., an encoder $\tilde{\mathbf{z}} = \psi_{\text{M}}^{\text{enc}}(\mathbf{z})$ and a decoder $\tilde{\mathbf{x}} = \psi_{\text{M}}^{\text{dec}}(\tilde{\mathbf{z}})$, both of which are implemented with MLPs.
- An underlying Mechanistic Neural Network, as detailed in the Lorenz experiment from Pervez et al. (2024). A mechanistic encoder inputs a trajectory and predicts the coefficients and parameters of the underlying ODE. The fast ODE solver of Chen et al. (2024) solves the ODE as a constrained optimization problem and predicts the trajectory from the initial condition, which is then matched to the data.

As a preprocessing step, the model unrolls trajectories with $t = 50$ steps per batch, resulting in batches of shape (B, t, D). The model is trained with the following loss:

$$\begin{aligned}\mathcal{L}_{\text{MNNAE}} = \left\|\mathbf{x} - \psi_{\text{M}}^{\text{enc}}\psi_{\text{M}}^{\text{dec}}(\mathbf{x})\right\|_2^2 + \beta_{\mathbf{z}} \left\|\tilde{\mathbf{z}} - \text{MNN}(\tilde{\mathbf{z}}(0))\right\|_2^2 + \\ + \beta_{\mathbf{x}} \left\|\mathbf{x} - \psi_{\text{M}}^{\text{dec}}(\text{MNN}(\tilde{\mathbf{z}}(0))\right\|_2^2 + \beta_1 \|\Xi\|_1,\end{aligned} \tag{45}$$

where each term represents, in order:

- The reconstruction of the autoencoder.
- The loss between the true encoded trajectory $\tilde{\mathbf{z}}$ and the one predicted by the MNN from the initial condition of the batch $\text{MNN}(\tilde{\mathbf{z}}(0))$.
- The loss between the true high-dimensional trajectory $\mathbf{x}$ and the decoded trajectory predicted by the MNN: $\psi_{\text{M}}^{\text{dec}}(\text{MNN}(\tilde{\mathbf{z}}(0)))$.
- The sparsity loss on Ξ.

Here, $\text{MNN}(\tilde{\mathbf{z}}(0)$ represents the solution of the ODE defined by the mechanistic encoder for the same number of steps as the batch, i.e., 50. Similar to SINDyAE, the equation is sparsified by thresholding coefficients below 0.1 every 100 epochs, for 10 times in total, as in Pervez et al. (2024). We use the original implementation available on GitHub.

Long Short-Term Memory. We implement an LSTM Hochreiter & Schmidhuber (1997) with linear readout layer and dropout with $p = 0.1$. To train the LSTM, we roll out the time series in batches of 50 steps. The LSTM receives as input the first n_{in} steps of the $\mathbf{x}$ batch, and predicts the following $50 - n_{\text{in}}$ steps autoregressively. We select the best model on the validation trajectories, choosing between $\{1, 2, 3\}$ layers and $n_{\text{in}} \in \{1, 10, 25, 49\}$ with the same batch size of SPEED-AE models. LSTMs are trained for 500 epochs with the ADAM optimizer and learning rate 0.001.

Transformer. We use a Transformer encoder model (Vaswani et al., 2017) with linear readout and positional encoding to predict the next steps of $\mathbf{x}$ from previous ones. Data is processed similarly to the LSTM model, where we roll out batches of trajectories of length 50. The Transformer receives as input the first n_{in} steps and autoregressively predicts the rest of the trajectory. We train the Transformer with the ADAM optimizer and cosine warmup scheduling for 500 epochs. We select the best model from $\{2, 3\}$ layers, $\{4, 8\}$ attention heads, and $n_{\text{in}} \in \{1, 10, 25, 49\}$.

D Details on the CRL methodologies

In this Section, we provide additional details on the CRL methods we used for the disentanglement step of our training pipeline. We remark that our work is not specifically tied to these methodologies, as any model capable of disentangling causal variables can be adopted.

D.1 CITRIS

CITRIS (Lippe et al., 2022) is a CRL approach designed for temporally intervened sequences, under the assumption that each true variable can be intervened upon in the data. While the method is designed for a more general setting with possibly multi-dimensional causal variables and minimal causal variables (i.e., the smallest set of variables that can be identified), in this work, we always assume that each variable can be intervened on at any time t. CITRIS needs to know the variable that has been intervened on at time t, that is, the intervention target $I_t \in \{0, 1, \dots, d\}$, where 0 means no intervention. For our purposes, we utilize CITRIS-NF, which consists of a pre-trained autoencoder that maps the images to a lower-dimensional space, followed by an invertible Normalizing Flow model that actually performs the disentanglement, as described in the original implementation by Lippe et al. (2022), which outputs the final latents $\hat{\mathbf{z}}^t = \hat{\mathbf{z}}(t)$. The model then consists of:

- a learnable assignment function $\psi : \{1, \dots, M\} \to \{0, \dots, K\}$, where, in our case, both M (the number of latent variables) and K (the number of causal blocks, together with an additional dimension for non-intervened ones) are set to d, the number of true latents,
- an autoregressive transition prior $p_{t1}(\hat{\mathbf{z}}^t | \hat{\mathbf{z}}^{t-1}, I^t)$, which is factorized as $p(\hat{\mathbf{z}}^t | \hat{\mathbf{z}}^{t-1}, I^t) = \prod_{i=1}^{d} p(\hat{z}^t_{\psi_i} | \hat{\mathbf{z}}^{t-1}, I^t_i)$, so that each varible $\hat{z}_i$ depends only on the past state and on its own intervention target,
- a target classifier, which is a small additional network trained to predict the intervention target I^{t+1} given the past and current state, which adds stability to the training procedure.

The overall loss is given by an ELBO calculated from the reconstruction error and the KL divergence on the transition prior, weighted by a hyperparameter β_{t1}, and the classification error from the target classifier, weighted by a hyperparameter $\beta_{\text{classifier}}$.

D.2 DMSVAE

Disentanglement via Mechanism Sparsity (Lachapelle et al., 2022; 2026) assumes that the causal graph between variables at two time steps is sparse, i.e., the graph G^z that represents the influences between $z_i(t-1)$ and $z_j(t)$ is sparse, or that the causal graphs G^a between some observed auxiliary variables $a_i(t)$ and the state $z_j(t)$ is sparse. Since in most dynamical systems the first assumption is very uncommon (usually, each ODE $\dot{z}_i = f_i(\mathbf{z}(t))$ involves almost every variable), we exploit the second one in a similar setting to CITRIS. In fact, we use the interventional targets $I(t)$ as observed auxiliary variables $a(t)$, which leads to a causal graph G^a being equal to the identity and, therefore, sparse. The identifiability comes from imposing such sparsity in the learned transition model. The full model (DMSVAE) consists of:

Table 2: Data generation parameters.

Parameter	Lotka-Volterra	Lorenz	Pendulum
n_{train}	20	1024	100
n_{val}	5	256	25
n_{test}	5	256	25
T	50	5	10
Δt	0.01	0.02	0.02
num. steps	5000	250	500
Sampled Variables	z_1, z_2	z_1, z_2, z_3	$z_1, \dot{z_1}, z_2, \dot{z_2}$
Sample distribution	Uniform $[0, 3]^2$	See Champion et al. (2019)	Uniform $[-2, 2]^4$
s_{train}	100000	256000	50000
s_{val}	25000	64000	12500
s_{test}	10000	25600	5000

- a variational autoencoder $\psi^{\text{enc}}_{\text{CRL}}, \psi^{[dec]}_{\text{CRL}}$,
- a transition model $p(\hat{z}_i(t)|\hat{\mathbf{z}}(t-1), I(t))$, parametrized by d MLPs,
- a causal graph $\hat{G}^a$, learned by binary masks applied on the transition model inputs, gating which variables it can use to predict the next one.

The model is trained with an ELBO loss based on the reconstruction error and the transition prior, weighted by β_{t1}, and a sparsity loss (the L^0 norm), on the causal graph $\hat{G}^a$, weighted by β_{sparse}.

E Details on the Experimental Setup

In this Section, we describe the complete setup of our experiments, from data generation to model selection and training, and provide the specific parameters and hyperparameters of each experiment.

Data Generation. In the experiments, we consider two kinds of trajectories: standard and interventional ones. To generate the first, we consider the ground truth ODE $\dot{\mathbf{z}} = \boldsymbol{f}(\mathbf{z})$ and sample $n_{\text{train}}, n_{\text{val}}, n_{\text{test}}$ initial conditions respectively for the training, validation, and test datasets. Each trajectory is calculated from the ODE and initial condition up to time T with time step Δt using the Runge-Kutta 45 solver from the SciPy library (Virtanen et al., 2020).

To generate the interventional trajectories, we instead iterate through the time dimension and, at each time step, choose a variable to intervene on with the same probability $p_{\text{intervention}} = 0.01$, or choose not to intervene with the remaining probability. If a variable i is selected, its next value $z_i(t + \Delta t)$ is randomly sampled, while the others evolve normally from $\mathbf{z}(t)$. If no variable is selected, all of them evolve normally. This is the same procedure used in CITRIS (Lippe et al., 2022), which we also adapt for DMSVAE (Lachapelle et al., 2022). Starting from a randomly sampled initial condition, we generate $s_{\text{train}}, s_{\text{val}}, s_{\text{test}}$ time steps for the training, validation, and test interventional trajectories, respectively. We provide the specific parameters of data generation in Table 2. To ensure that the standard trajectories are in-distribution for the CRL models, we first generate the interventional ones, and then generate standard ones until we have enough withing within the bounds of the interventional data. Finally, all trajectories $\mathbf{z}(t)$ are mapped to the higher-dimensional ones $\mathbf{x}(t)$ via an experiment-specific mapping, which we describe in the following appendices.

CRL model training. To obtain disentangled representations $\hat{\mathbf{z}}$, we use CITRIS (Lippe et al., 2022) or DMSVAE (Lachapelle et al., 2022), which we discuss in Appendix D. After training, CRL models have learned an encoder $\hat{\mathbf{z}} = \psi^{\text{enc}}_{\text{CRL}}(\mathbf{x})$ as well as a permutation π that assigns the true latent z_i to the corresponding found latent $\hat{z}_{\pi(i)}$. In general, such an optimal permutation can be found via the best mean correlation coefficient across permutations. On validation trajectories, the R^2 between each z_i and the corresponding $\hat{z}_{\pi(i)}$ is calculated, and the average across the variables is taken. We select the model with the highest statistic from a simple grid search on hyperparameters with values in $\{0.001, 0.01, 0.1, 1.0, 10.0\}$. We report the best combinations in Table 3. Details on

Table 3: CRL model hyperparameters. Lotka-Volterra with $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$ values are not reported, as in this experiment, we use the same pre-trained CRL models coming from $\alpha = \frac{2}{3}, \beta = \frac{4}{3}$.

CITRIS					
Hyperparameter	Lotka-Volterra $\alpha = \frac{2}{3}, \beta = \frac{4}{3}$	Lotka-Volterra $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$	Lorenz (Stable) $\rho = 14$	Lorenz (Chaotic) $\rho = 28$	Pendulums
β_{t1}	10	/	0.01	0.1	0.01
$\beta_{\text{classifier}}$	1	/	0.001	0.01	0.001
DMSVAE					
Hyperparameter	Lotka-Volterra $\alpha = \frac{2}{3}, \beta = \frac{4}{3}$	Lotka-Volterra $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$	Lorenz (Stable) $\rho = 14$	Lorenz (Chaotic) $\rho = 28$	Pendulums
β_{t1}	1.0	/	1.0	1.0	/
β_{sparse}	0.01	/	0.01	0.01	/

the architectures for each experiment are provided in the following Appendices. The setup for both models is the same as in Lippe et al. (2022; 2023).

We train the CRL methods on every dataset, except for Lotka-Volterra with $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$, where we used the one from the other Lotka-Volterra experiment to show representation transferability.

SPEED-AE training Once the CRL model is trained, we consider the high-dimensional standard trajectories $\mathbf{x}$ we generated (with no interventions), and encode them through the CRL model $\hat{\mathbf{z}} = \psi_{\text{CRL}}^{\text{enc}}(\mathbf{x})$. Then, we permute the latents following the learned assignments π, so that we can assume $\hat{z}_i$ corresponds to z_i with $\pi(i) = i$. Then, for the SINDy-based versions of SPEED-AE, we calculate the derivatives of these trajectories using second-order finite differences. For SPEED-AE models with the MNN-based loss, we instead unroll trajectories with $t = 50$ steps per batch, similar to Pervez et al. (2024), obtaining batches with shape (B, t, d). Then, model selection is performed on the hyperparameters of each version of SPEED-AE: $\beta_{\mathbf{z}}$ and $\beta_{\mathbf{x}}$ are chosen from $\{10^{-5}, 10^{-4}, 0.001, 0.01, 0.1, 1.0, 10.0\}$, while β_1 is chosen from $\{0.001, 0.0001, 0.00001\}$. Final values are reported in Table 4. To select the best model, we consider forecasting performance from the initial condition on validation trajectories. In particular, we first encode the validation trajectories through the CRL model $\hat{\mathbf{z}} = \psi_{\text{CRL}}^{\text{enc}}(\mathbf{x})$ and the SPEED-AE one $\tilde{z}_i(0) = \phi_i^{\text{enc}}(\hat{z}_i(0))$. Then, the trajectory is predicted using the Runge-Kutta 45 solver on the learned ODE $\dot{\tilde{\mathbf{z}}} = \Theta(\tilde{\mathbf{z}})^\top \Xi$. Finally, the whole trajectories are decoded via the component-wise decoders ϕ_i^{dec} and compared to the original values of $\hat{\mathbf{z}}$. Table 4 reports the best combinations for each SPEED-AE model, which depend on the CRL model as well. In each case, the component-wise encoders and decoders are MLPs with 2 layers of 64 units each. The models are trained with the ADAM optimizer (Kingma & Ba, 2017) with a learning rate of 0.0005 for the SINDy-based loss and 0.0001 for the MNN-based one. The models are trained for 1000 epochs with sequential thresholding: every 100 epochs, coefficients Ξ_i below 0.1 are zeroed and masked. The batch size is chosen so that each epoch consists of around 100 steps and is reported in Table 4 for each experiment.

SINDyAE baseline. The SINDyAutoencoder model implementation is taken directly from Champion et al. (2019). Training follows the same procedure as in their original work. The loss weights are chosen in $\{10^{-5}, 10^{-4}, 0.001, 0.01, 0.1, 1.0, 10.0\}$ for the Lotka-Volterra and Pendulums experiments, while we use the authors' parameters for Lorenz, since the standard trajectory generation is the same. To select the best model, we first encode the validation trajectories $\mathbf{x}(t)$ in the learned representation space $\tilde{\mathbf{z}}(t)$. We calculate a prediction of these trajectories from the initial condition $\tilde{\mathbf{z}}(0)$ and the learned ODE, obtaining $\tilde{\mathbf{z}}_{\text{pred}}(t)$, and we decode into the predicted $\mathbf{x}_{\text{pred}}(t)$. Finally we calculate the L^2 error between these $\mathbf{x}_{\text{pred}}$ and $\mathbf{x}_{\text{true}}$. This measures how well the learned ODEs describe the encoded data. Models consist of MLPs with the same architecture as SPEED-AE for direct comparability. The model is trained with the ADAM optimizer with learning rate 0.0001 (as in Champion et al. (2019)), and the same batch size as SPEED-AE. Models are trained for 5000 epochs with thresholding every 500, and 1000 epochs of refinement (same as Champion et al. (2019)).

MNNAE baseline. The implementation of the MNN is the one of Pervez et al. (2024); Chen et al. (2024), where the underlying MNN is coupled with an Autoencoder, as detailed in Appendix C. The

Table 4: SPEED-AE models hyperparameters.

SPEED-AE (CITRIS + SINDy-based loss)					
Hyperparameter	Lotka-Volterra $\alpha=\frac{2}{3},\beta=\frac{4}{3}$	Lotka-Volterra $\alpha=\frac{6}{5},\beta=\frac{3}{2}$	Lorenz (Stable) $\rho=14$	Lorenz (Chaotic) $\rho=28$	Pendulums
$\beta_{\mathbf{x}}$	0.01	0.01	0.01	0.01	0.01
$\beta_{\mathbf{z}}$	0.001	0.0001	0.001	0.0001	0.0001
β_1	0.0001	0.0001	0.00001	0.0001	0.0001
batch_size	4096	4096	2048	2048	512
SPEED-AE (DMSVAE + SINDy-based loss)					
Hyperparameter	Lotka-Volterra $\alpha=\frac{2}{3},\beta=\frac{4}{3}$	Lotka-Volterra $\alpha=\frac{6}{5},\beta=\frac{3}{2}$	Lorenz (Stable) $\rho=14$	Lorenz (Chaotic) $\rho=28$	Pendulums
$\beta_{\mathbf{x}}$	0.01	0.01	0.001	0.001	/
$\beta_{\mathbf{z}}$	0.001	0.0	0.01	0.001	/
β_1	0.00001	0.00001	0.001	0.00001	/
batch_size	4096	4096	2048	2048	/
SPEED-AE (CITRIS + MNN-based loss)					
Hyperparameter	Lotka-Volterra $\alpha=\frac{2}{3},\beta=\frac{4}{3}$	Lotka-Volterra $\alpha=\frac{6}{5},\beta=\frac{3}{2}$	Lorenz (Stable) $\rho=14$	Lorenz (Chaotic) $\rho=28$	Pendulums
$\beta_{\mathbf{x}}$	0.01	0.1	0.01	0.01	/
$\beta_{\mathbf{z}}$	10.0	10.0	10.0	1.0	/
β_1	0.001	0.001	0.001	0.0001	/
batch_size	4096	4096	2048	2048	/
SPEED-AE (DMSVAE + MNN-based loss)					
Hyperparameter	Lotka-Volterra $\alpha=\frac{2}{3},\beta=\frac{4}{3}$	Lotka-Volterra $\alpha=\frac{6}{5},\beta=\frac{3}{2}$	Lorenz (Stable) $\rho=14$	Lorenz (Chaotic) $\rho=28$	Pendulums
$\beta_{\mathbf{x}}$	0.01	0.01	0.01	0.1	/
$\beta_{\mathbf{z}}$	10.0	0.1	1.0	1.0	/
β_1	0.00001	0.00001	0.00001	0.00001	/
batch_size	4096	4096	2048	2048	/

loss weights are chosen among $\{0.001, 0.01, 0.1, 1.0, 10.0\}$ for $\beta_{\mathbf{z}}$, $\beta\mathbf{z}$ and $\{0.001, 0.0001, 0.00001\}$ for β_1. The model selection procedure follows that of SINDyAE. The architecture is the same as SPEED-AE for direct comparability. Finally, MNNAE is trained with the ADAM optimizer and learning rate 0.0005, with batch size, number of epochs, and sequential thresholding as those of SPEED-AE.

Evaluation. The final evaluation is performed on the test trajectories following the evaluation pipeline in Appendix F.

E.1 Hardware and Computational Costs

For all experiments, we use a machine with NVIDIA H100 GPUs, and each model has access to one GPU and at most 2 CPUs. We report the training time for the Lotka-Volterra experiment, for reference, in Table 5. Thus, the full SPEED-AE pipeline (including disentanglement), requires approximately the same or less time than the other two baselines (SINDyAE and MNNAE).

F Evaluation Pipeline

In this Section, we detail the two protocols used for evaluating the performance of each model in terms of ODE discovery and trajectory prediction. A detailed graphical representation is provided in Figure 5, which we use in the next paragraphs to describe the protocols. The green boxes represent the true latent variables $\mathbf{z}$ and the ground truth ODEs $\dot{\mathbf{z}} = \boldsymbol{f}(\mathbf{z})$, which are kept unknown during training and used only during evaluation. Models have access only to the images/high-dimensional data $\mathbf{x}(t)$ in the top-left corner, while the blue boxes on the right represent their outputs, namely the encodings $\tilde{\mathbf{z}}(t)$ and the recovered ODEs in terms of $\tilde{\mathbf{z}}$.

Each model learns a specific set of latent variables, e.g. $\tilde{\mathbf{z}}_{\text{SINDyAE}}, \tilde{\mathbf{z}}_{\text{SPEED-AE(D+S)}}$, which live in different spaces, each with its own ODE, making them not directly comparable. To evaluate them

Table 5: Training time for each model on the Lotka-Volterra experiment.

Model	Training Time (mins)
LSTM	≈ 45
Transformer	≈ 62
SINDyAE	≈ 100
MNNAE	≈ 191
Autoencoder (CITRIS-NF)	≈ 22
CITRIS-NF	≈ 52
DMSVAE	≈ 41
SPEED-AE SINDy-based	≈ 57
SPEED-AE MNN-based	≈ 39

on a common ground, our approach consists of selecting a representative latent space, where the encoded trajectories and recovered ODEs are converted. Since we want to stay unbiased towards which methodology should be preferred, we make the most natural choice possible, which consists of using the true latent space of $\mathbf{z}$ as a common evaluation space. Since the true latent variables $\mathbf{z}$ are used only during this evaluation, the models will have no access to them before this phase.

Evaluation protocol: Correlation Discrepancy. We measure how well the encoded data $\tilde{\mathbf{z}}$ is correlated to the true latents $\mathbf{z}$ by comparing the true Pearson's correlation coefficient matrix $\rho_{ij} = \frac{\text{cov}(z_i, z_j)}{\sigma_{z_i}\sigma_{z_j}}$ and its cross version between true and found latents $\tilde{\rho}_{ij} = \frac{\text{cov}(z_i, \tilde{z}_j)}{\sigma_{z_i}\sigma_{\tilde{z}_j}}$. Once these matrices are calculated on the train or test trajectories, we compare the two matrices by averaging the absolute error (Correlation Discrepancy) $\text{CorrD} = \frac{1}{N^2}\sum_{i,j=1}^{N} |\rho_{ij} - \tilde{\rho}_{ij}|$.

Evaluation protocol: time-series prediction. With the second set of metrics, we evaluate how well the models and, consequently, the learned ODEs, can predict the long-term behavior of the dynamical systems from just their initial conditions. Hence, we let each model encode the given initial condition $\mathbf{x}(0)$ and evolve it via the dynamics it has learned, obtaining the full trajectory $\tilde{\mathbf{z}}_{\text{pred}}(t)$. One natural metric to calculate is the error in the final prediction of $\mathbf{x}(t)$, obtained by decoding the predicted trajectory in the latent space onto the higher-dimensional space. However, to gain insight into the learned latents and their dynamics, we also want to evaluate them directly against the true ones $\mathbf{z}(t)$. As above, we need to evaluate the models on a reference space, that is, the space of true latents $\mathbf{z}$. However, here we do not need to stick to polynomial mappings and transformations, as we instead care solely about the predictive capabilities. Hence, we learn an oracle MLP for each model that converts the model latents $\tilde{\mathbf{z}}_{\text{pred}}(t)$ into the true ones. Then, the time-averaged L^2 distance

$$\frac{1}{n_{\text{steps}}} \sum_{i=1}^{n_{\text{steps}}} \|\text{MLP}(\tilde{\mathbf{z}}(t_i)) - \mathbf{z}(t_i)\|_2^2 \tag{46}$$

or any other metric between true and converted latents can be calculated. While CRL models provide an assignment between true and found latents, for SINDyAE and MNNAE we consider the permutation given by the best Mean Correlation Coefficient, as described above.

Evaluation protocol: ODE discovery evaluation Now that we have set the reference space for evaluation, we define the exact protocol to transform the discovered ODEs from the latent space of the model $\mathcal{Z}_{\text{model}}$ to the true latent space $\mathcal{Z}$. We work with the simplifications used previously, and hence we assume that the two spaces are separated by a component-wise polynomial transformation, that is $\tilde{z}_i = \text{Poly}_{i,\text{model}}(z_{i,\text{model}})$. This is helpful to convert the ODEs between one space and the other, preserving a polynomial representation. Furthermore, on compact spaces, polynomial functions are a complete basis and can therefore approximate any map between learned and true latents. This polynomial transformation $\tilde{z}_i = \text{Poly}_{i,\text{model}}(z_{i,\text{model}})$ is calculated via linear regression on polynomial features starting from degree 1 and up to degree 3, which we found to be enough in all cases, stopping when the score is higher than 0.99 (purple box). This greedy approach ensures that the learned transformation is both accurate and simple, using higher-degree polynomials only

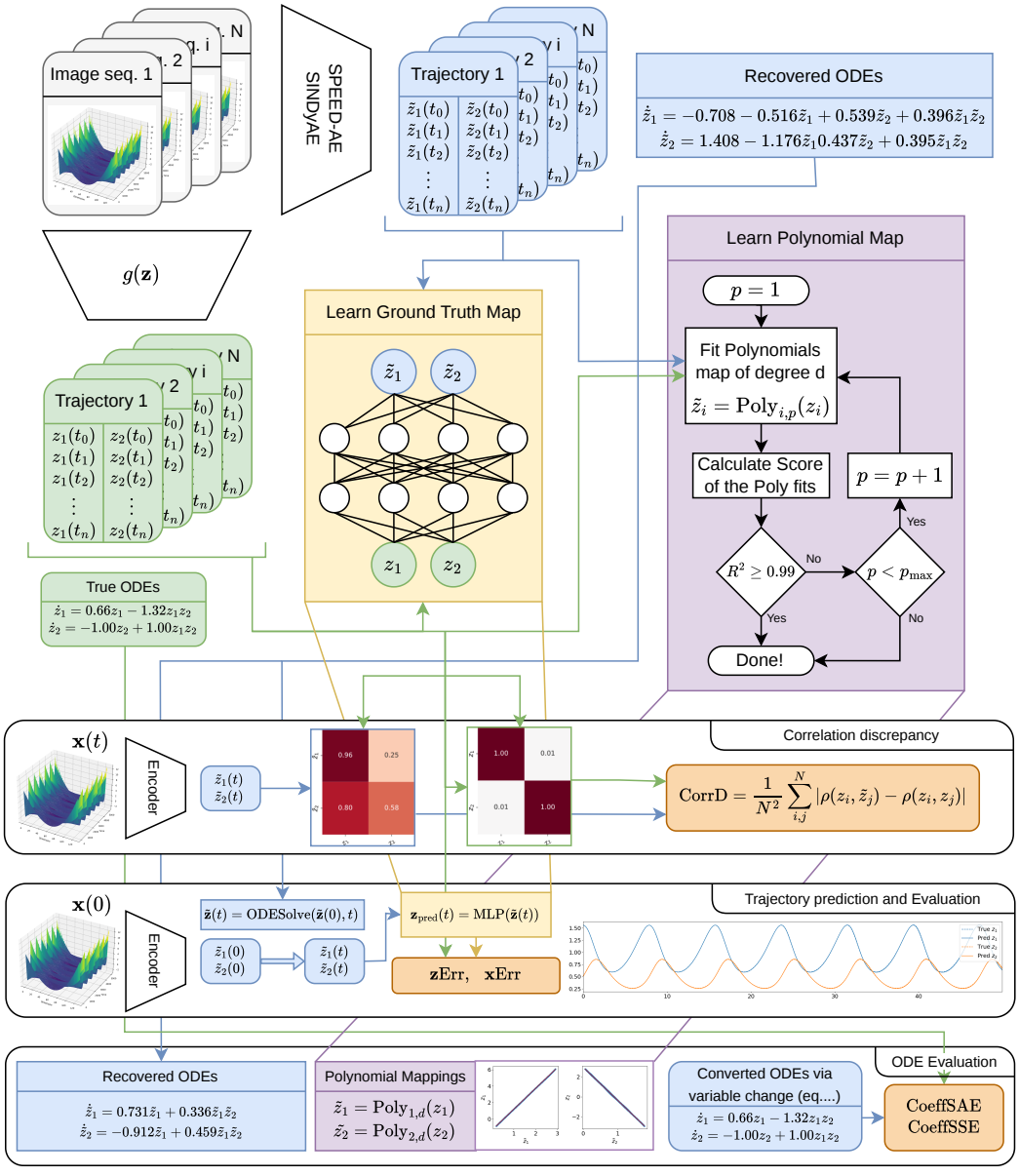


Figure 5: Evaluation pipeline of our experiments.

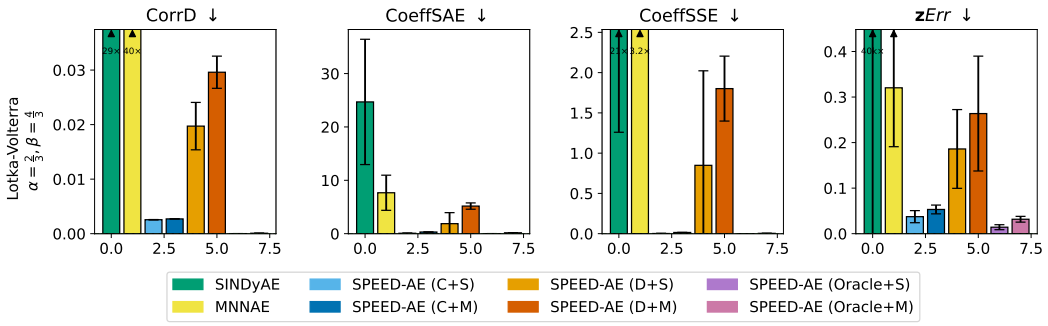


Figure 6: Experimental results for the ablation on the independence from CRL method. Lotka-Volterra dataset with an additional version of SPEED-AE with an oracle CRL model that achieves perfect disentanglement and SINDy-based loss.

if necessary. Otherwise, the transformed ODE would present more terms. Finally, the found ODE function $\dot{\tilde{\mathbf{z}}} = \tilde{f}(\tilde{\mathbf{z}})$ (blue box in the bottom rectangle) is transformed via a change of variable with the fitted polynomial mapping (purple box), obtaining an ODE written in terms of the true latent variables $\dot{\mathbf{z}} = f_{\text{model}}(\mathbf{z})$ (blue box). Since we always considered a polynomial library and polynomial transformations, this new ODE $f_{\text{model}}(\mathbf{z})$ will necessarily be polynomial, possibly in implicit form. Before calculating the error, we remove coefficients below the 0.001 threshold, which do not influece the results and provide a cleaner ODE representation. This ODE can be easily compared to the true one $f(\mathbf{z})$ by looking at their coefficients Ξ and Ξ_{model} when expressed in library form. Practically, we consider the sum of absolute $\|\Xi - \Xi_{\text{model}}\|_1$ and squared $\|\Xi - \Xi_{\text{model}}\|_2^2$ values of their difference as a measure of their distance. In case an ODE is implicit, we also consider all terms $z^\ell \dot{z}_i$ in the library. Since these terms do not appear in the true ODE, we penalize models that find an ODE equivalent to an implicit one. Our theoretical discussion shows that implicit terms appear if and only if the map is monomial or polynomial with degree greater than one instead of linear, and we want to penalize modes that cannot find such simple maps and reward those that do. This way, we reward those modes that are more interpretable and achieve a better identifiability result.

SINDyAE and MNNAE present a particular challenge within this framework, as they do not provide a valid variable assignment between real and found latents. Furthermore, disentanglement is not guaranteed. To solve this issue, we do the following: we first calculate the Mean Correlation Coefficient (MCC) among all possible permutations, hence finding the best variable assignment $z_i = \tilde{z}_{\pi(i)}$. Then, we proceed as above from the polynomial mapping step.

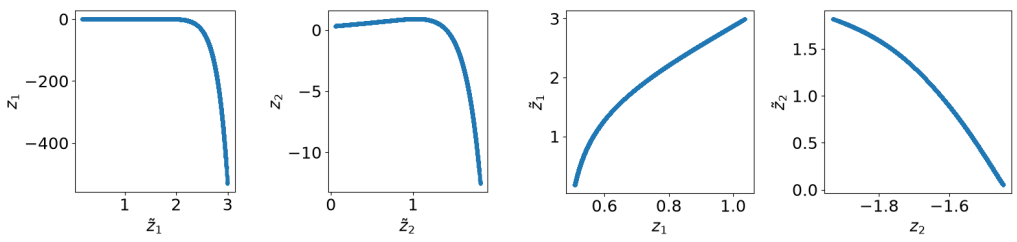
G Ablations

Independence from the CRL method. We isolate the main component of SPEED-AE, that is, the component-wise autoencoder coupled with sparse ODE recovery, to show that it is independent of the CRL method used. To do so, we assume an oracle CRL method that achieves perfect disentanglement and identifies the true latents up to permutation and component-wise diffeomorphism. To implement this, we first choose random coefficients $\alpha_{i,j} \in (0, 1)$ with $i = 1, \dots, d$ and $j = 0, 1, 3$, defining the polynomials

$$P_i(z_i) = \alpha_{i,0} + \alpha_{i,1}(z_i + 1) + \alpha_{i,3}(z_i + 1)^3, \tag{47}$$

and the oracle CRL variables as $\hat{z}_i = P_i(z_i)$. We use $(z_i + 1)$ to (i) offset the variables from their original values, and (ii) have a polynomial mapping that also has a z_i^2 term. Given the form of Equation 47, their derivatives are always positive because they include only even-power terms of $(z_i + 1)$ with positive coefficients, guaranteeing monotonicity and thus invertibility.

We consider the Lotka-Volterra experiment in Section 6. We calculate the $\hat{\mathbf{z}}$ variables directly from the true latents $\mathbf{z}$ during data preparation. The rest of the experiment proceeds as described in Appendix E, starting with SPEED-AE training on the latents $\hat{\mathbf{z}}$. We remark that SPEED-AE never has access to the true latents $\mathbf{z}$.



(a) (b)

Figure 7: Results on the synthetic experiments with perfect disentanglement, polynomial identifiability in the CRL latents, and SPEED-AE encoder and decoders implemented as polynomials. (a) Some of the runs show strong instabilities, while others (b) remain stable but with flat dynamics. In general, the polynomial functions are hard to optimize simultaneously to the dynamics.

Results are reported in Figure 6. SPEED-AE with the SINDy-based loss (Oracle+S) and MNN-based one (Oracle+M) achieve the lowest errors across every metric. Thus, SPEED-AE can virtually achieve almost perfect results if the disentanglement is good enough. We notice that the MNN-based model has slightly worse performance, which is also reflected in the real results, where SINDy-based combinations of SPEED-AE are often better.

Polynomial diffeomorphisms. Inspired by our discussion in Section 3 and Appendix A, we first experimented with a fully polynomial setting. In particular, we ran synthetic experiments similar to the ones above, with perfect disentanglement and identifiability up to polynomial transformation. The only difference is in the SPEED-AE encoders and decoders $\phi^{\text{enc}}, \phi^{\text{dec}}$, which we implemented as polynomial functions. In such a case, we would have the guarantee that the new latents z_i are a polynomial transformation of the found ones, as $z_i = \hat{h}_i \circ \phi_i^{\text{dec}}(\tilde{z}_{\pi(i)})$, where both ϕ_i^{dec} and $z_i = \hat{h}(\hat{z}_{\pi(i)})$, the diffeomorphism between z_i and $\hat{z}_i$, are both polynomial.

Unfortunately, the results were not satisfactory. Using polynomial functions as diffeomorphisms leads to strong instabilities in the model, and we observed either blow-ups in the solutions or flat dynamics. We also tried to simplify the setting by considering two simple oracle CRL variables: $\hat{z}_i = a_i \sqrt[3]{z_i}$, for which the polynomial encoder would only need to learn to use the cubic power term, and $\hat{z}_i = a_i z_i$, i.e., the CRL variables are already an affine transformation of the true ones. Even in these cases, we were not able to effectively train the models, as the polynomial functions make it difficult to train the autoencoder and the dynamics simultaneously. We report some plots in Figure 7. As a consequence, we use MLPs to implement the encoder and decoders.

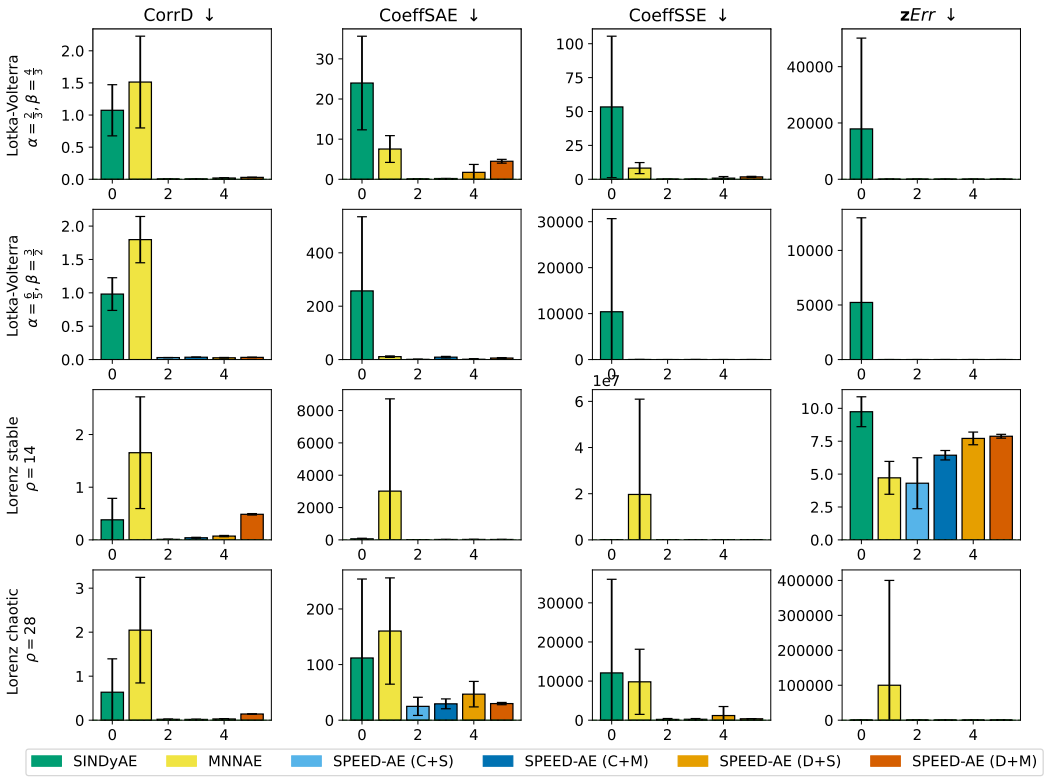


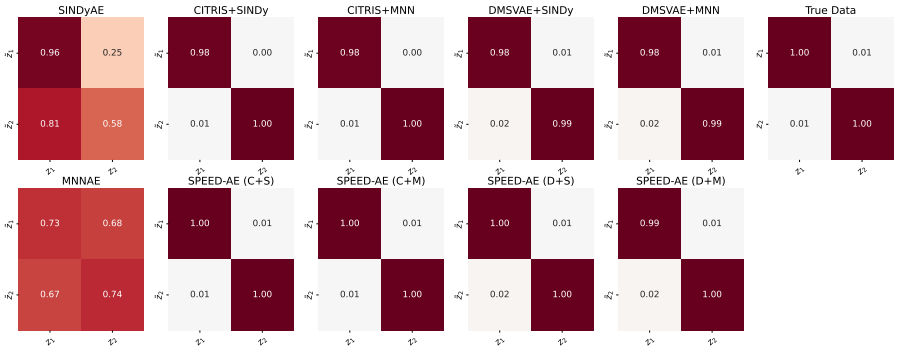
Figure 8: Experimental results for the Lotka-Volterra and the Lorenz experiments.

H Additional Experimental Results

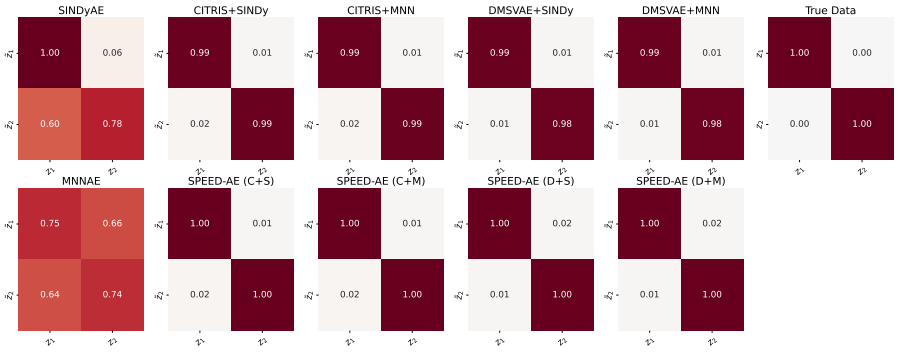
We report the uncut version of Figure 4 in Figure 8. In many plots, especially those related to the Correlation Discrepancy and errors in the Recovered ODE coefficients, SINDyAE and MNNAE are out of scale compared with all combinations of SPEED-AE.

Forecasting error on $\mathbf{x}$. We report the results on the $\mathbf{x}$Err metric in Table 6, with the additional baselines of LSTM and Transformer. Traditional ML baselines perform well on the short trajectories of the Lorenz experiment (250 steps), even in the chaotic case. However, on the longer trajectories of the Lotka-Volterra experiment (5000 steps), the predictions collapse (see Figure 13 and Figure 14) and the errors are almost 20 times higher than SPEED-AE. In general, MNNAE performs well across experiments, although it fails on the chaotic case of Lorenz. However, as already discussed in Section 6, this performance comes at the cost of no identifiability of the latents.

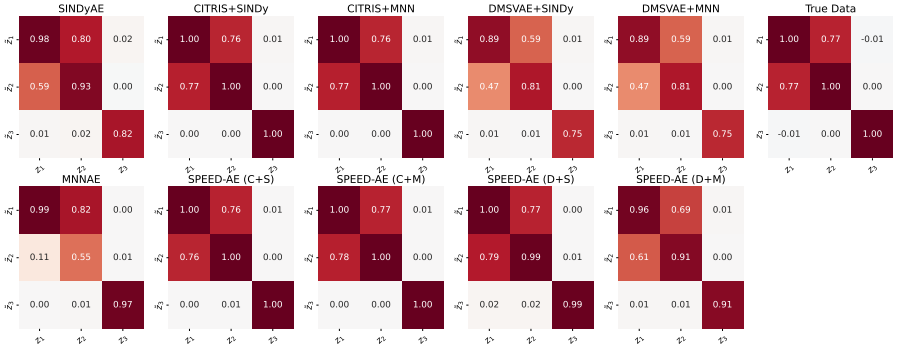
In the following results, we always permute the recovered latents to match the true ones so that, for the ease of readability, $\pi(i) = i$. We remark that CRL latents identify the true ones up to permutation.



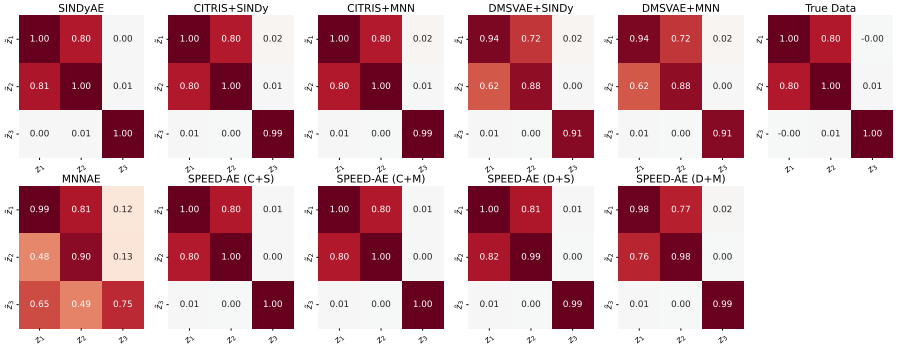
(a) Lotka-Volterra with $\alpha = \frac{2}{3}, \beta = \frac{4}{3}$



(b) Lotka-Volterra with $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$



(c) Lorenz stable with $\rho = 14$



(d) Lorenz chaotic with $\rho = 28$

Figure 9: Pearson cross correlation matrices between $\tilde{\mathbf{z}}$ and $\mathbf{z}$ for all experiments and models. The rightmost matrix in each subfigure corresponds to the true correlation matrix (i.e., between $\mathbf{z}$ and $\mathbf{z}$).

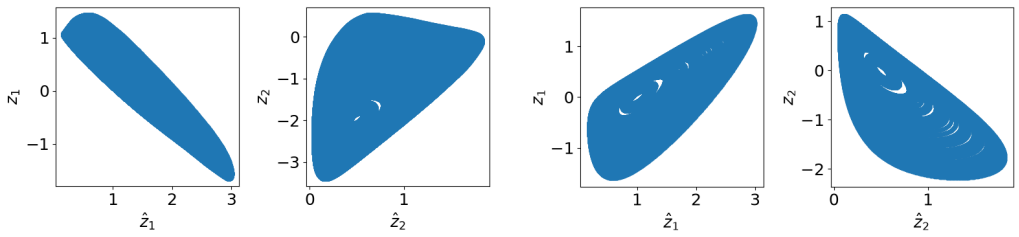
Table 6: $\mathbf{x}$Err: mean and standard deviation across 10 seeds.

Model	Lotka-Volterra $\alpha = \frac{2}{3}, \beta = \frac{4}{3}$	Lotka-Volterra $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$	Lorenz (Stable) $\rho = 14$	Lorenz (Chaotic) $\rho = 28$
LSTM	$13.860_{\pm 2.609}$	$29.313_{\pm 14.719}$	$0.157_{\pm 0.056}$	$1.624_{\pm 0.210}$
Transformer	$16.001_{\pm 2.076}$	$25.999_{\pm 6.919}$	$0.422_{\pm 0.076}$	$1.902_{\pm 0.210}$
SINDyAE	$23.075_{\pm 11.611}$	$17.610_{\pm 2.898}$	$1.988_{\pm 0.323}$	$3.137_{\pm 0.120}$
MNNAE	$1.257_{\pm 0.240}$	$0.930_{\pm 0.290}$	$0.448_{\pm 0.102}$	$\approx 10^5$ (failed)
SPEED-AE (C+S)	$1.279_{\pm 0.483}$	$3.999_{\pm 2.390}$	$0.851_{\pm 0.384}$	$2.411_{\pm 0.379}$
SPEED-AE (C+M)	$1.552_{\pm 0.211}$	$16.041_{\pm 4.801}$	$1.256_{\pm 0.066}$	$2.984_{\pm 0.215}$
SPEED-AE (D+S)	$4.669_{\pm 2.074}$	$4.881_{\pm 2.363}$	$1.399_{\pm 0.036}$	$2.736_{\pm 0.509}$
SPEED-AE (D+M)	$7.239_{\pm 3.442}$	$13.686_{\pm 5.811}$	$1.619_{\pm 0.039}$	$2.684_{\pm 0.260}$

H.1 Lotka-Volterra Experiment

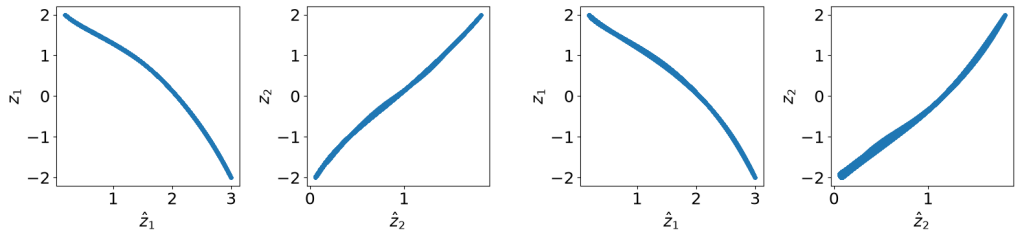
To visualize how the learned latents of each model $\tilde{\mathbf{z}}$ are related to the true ones $\mathbf{z}$, we scatter plot them for the whole dataset in Figure 10 and Figure 11. In both Lotka-Volterra experiments, SINDyAE and MNNAE do not disentangle the variables, as there is no one-to-one correspondence between each $\tilde{z}_i$ and z_i. Similarly, their correlation matrices in Figure 9a and Figure 9b are very different from the true one. On the other hand, CITRIS and DMSVAE successfully disentangle the variables, but the mapping is not linear. SPEED-AE improves on this in almost every case, indicating that the model achieves identifiability up to linear transformation. We also report the recovered ODE coefficients in the true latent space $\mathbf{z}$ for the Lotka-Volterra experiment with transfer of CRL representations and $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$. A similar conclusion to the results in Section 6 holds even in this case: SPEED-AE recovers the sparsest and most similar equations, while SINDy and MNN applied to CRL latents have to sacrifice sparsity and interpretability for good forecasting.

Finally, we report representative plots of $\mathbf{x}$ forecasting (first 10 components out of 128) for the employed ML baselines in Figure 13 and Figure 14. In both cases, model rollout trajectories collapse in less than 1000 steps.



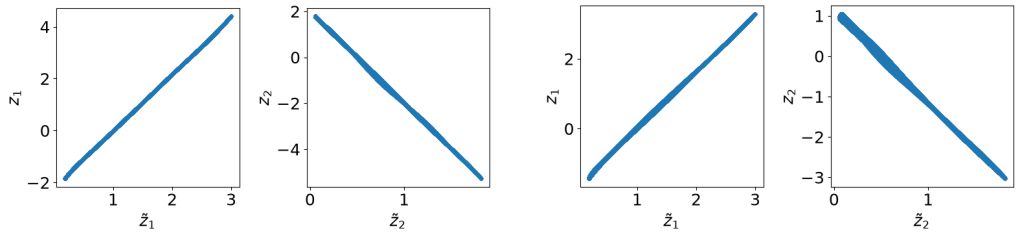
(a) SINDyAE

(b) MNNAE



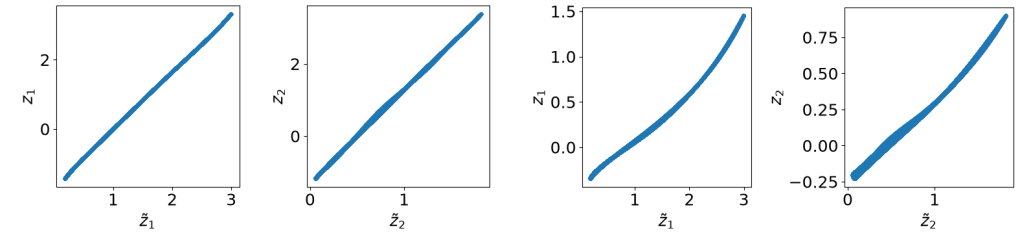
(c) CITRIS

(d) DMSVAE



(e) SPEED-AE (C+S)

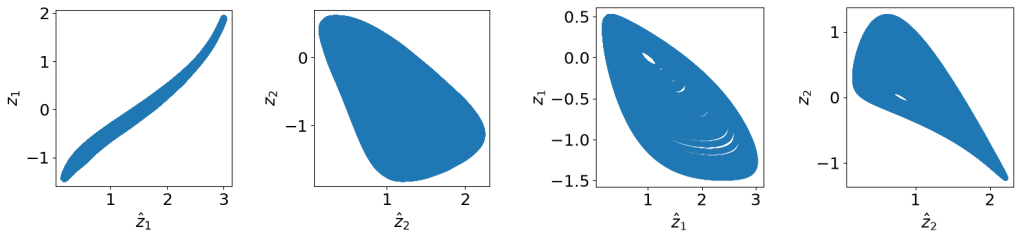
(f) SPEED-AE (D+S)



(g) SPEED-AE (C+M)

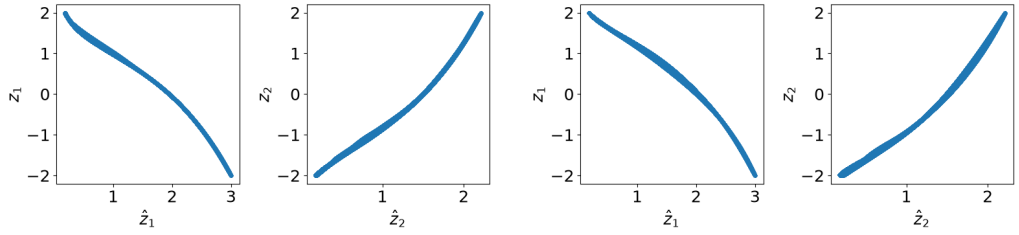
(h) SPEED-AE (D+M)

Figure 10: Lotka-Volterra experiment with $\alpha = \frac{2}{3}, \beta = \frac{4}{3}$. Scatter plots of the found latents $\tilde{\mathbf{z}}$ against the true ones $\mathbf{z}$. If their function is well defined, the plots well represent the maps $z_i = h_i(\tilde{z}_i)$. In the second row, we include the same plots for the CRL latents $\hat{\mathbf{z}}$ from CITRIS and DMSVAE. In most cases, SPEED-AE can go from a general diffeomorphism h to a linear one.



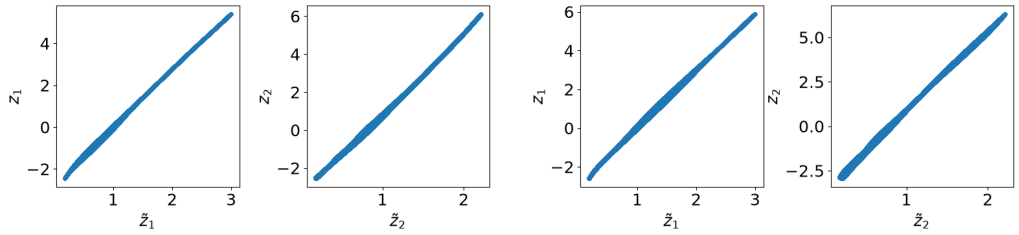
(a) SINDyAE

(b) MNNAE



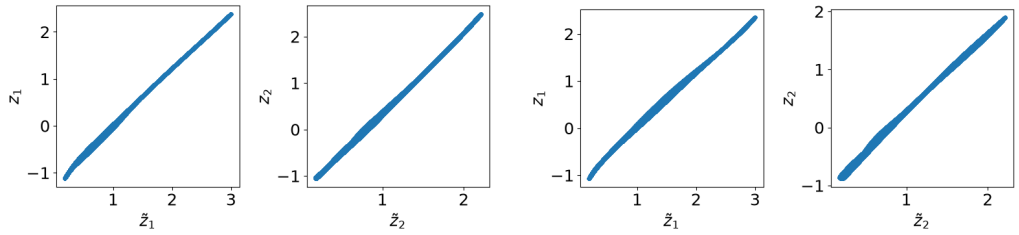
(c) CITRIS

(d) DMSVAE



(e) SPEED-AE (C+S)

(f) SPEED-AE (D+S)



(g) SPEED-AE (C+M)

(h) SPEED-AE (D+M)

Figure 11: Lotka-Volterra experiment woth $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$. Scatter plots of the found latents $\tilde{\mathbf{z}}$ against the true ones $\mathbf{z}$. If their function is well defined, the plots well represent the maps $z_i = h_i(\tilde{z}_i)$. In the second row, we include the same plots for the CRL latents $\hat{\mathbf{z}}$ from CITRIS and DMSVAE. In most cases, SPEED-AE can go from a general diffeomorphism h to a linear one.

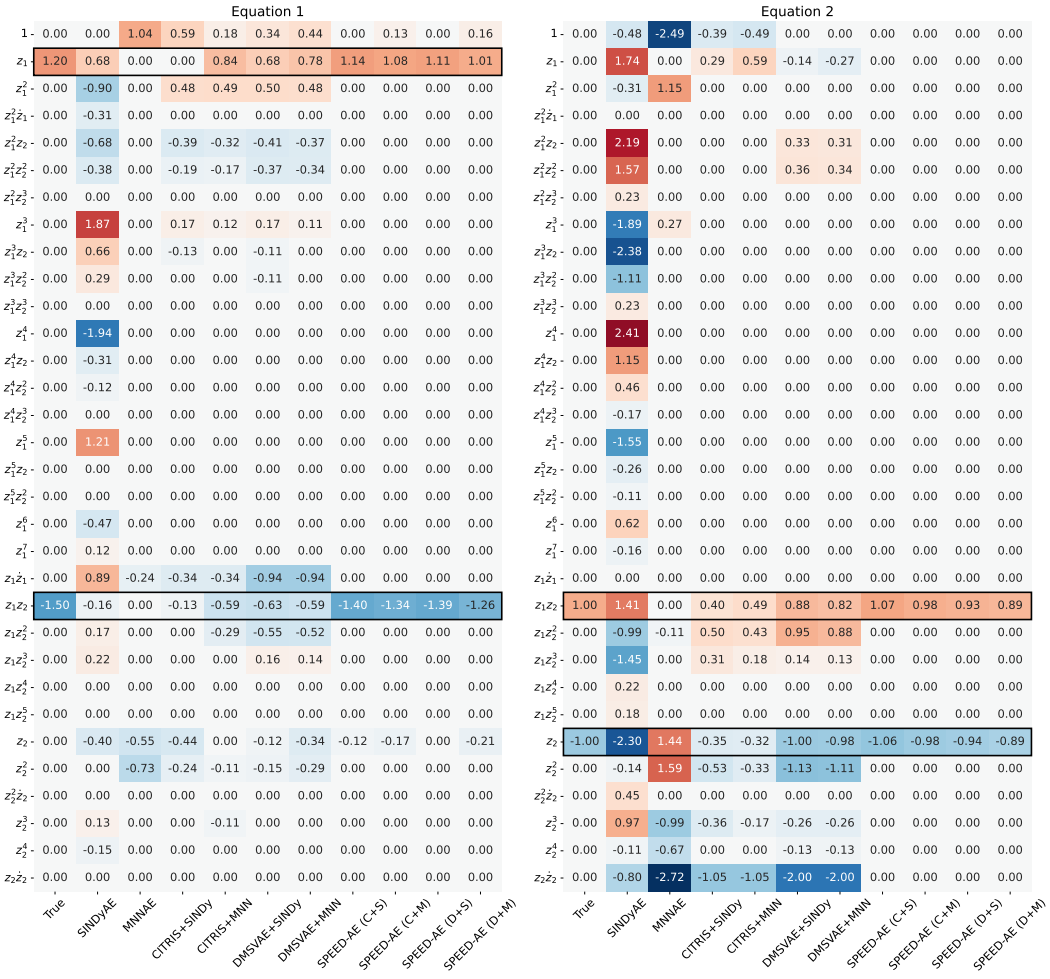


Figure 12: Identified coefficients for the Lotka-Volterra experiment with transfer of CRL models and $\alpha = \frac{6}{5}, \beta = \frac{3}{2}$. The true ones are in the leftmost column and highlighted by the black borders. For comparison, we include four baselines that consist of equation discovery directly applied to the CRL latents $\hat{\mathbf{z}}$.

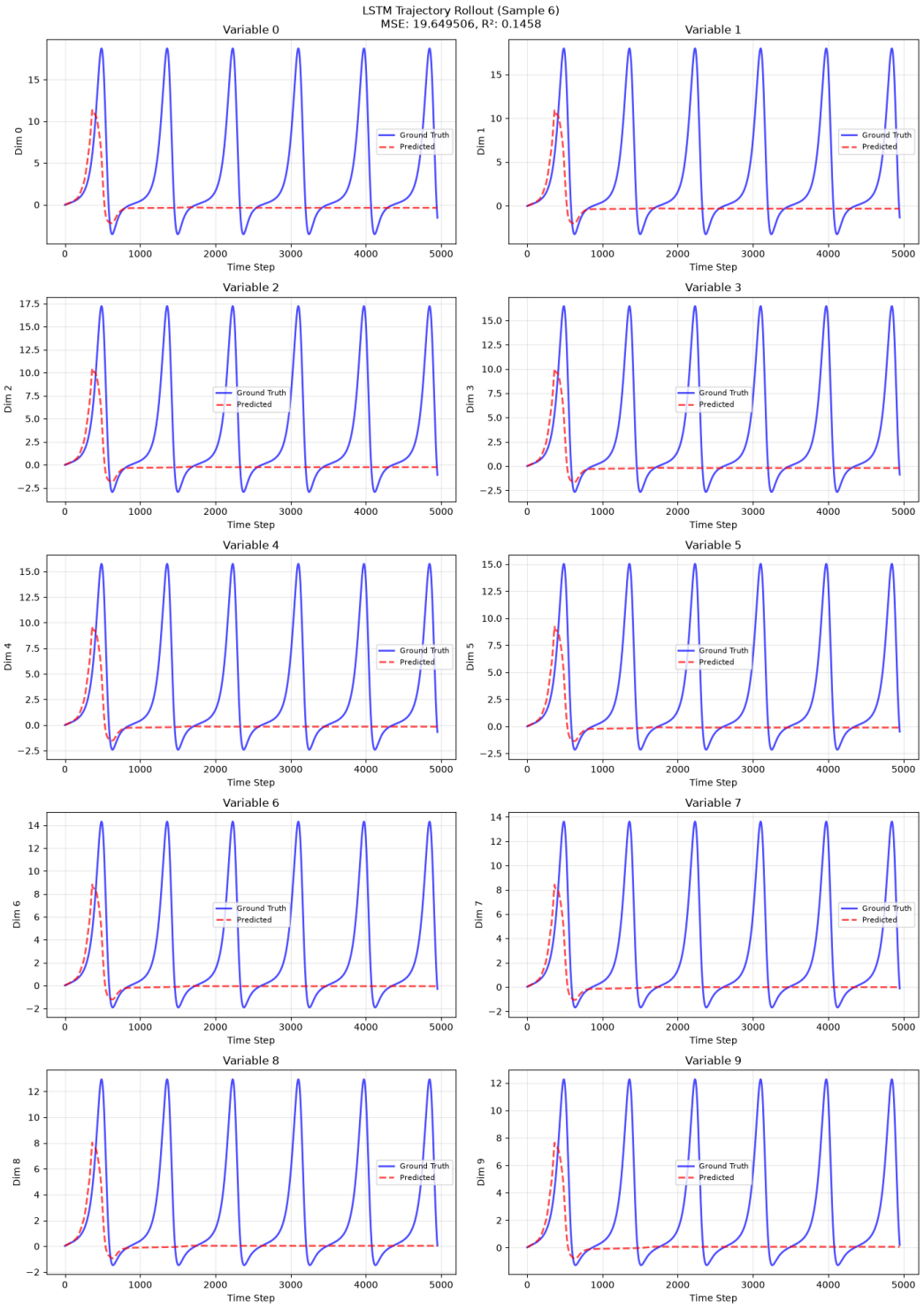


Figure 13: Example test trajectory of the Lotka-Volterra experiment (first 10 dimensions of $\mathbf{x}$). Ground truth (blue) against the predicted trajectory from the LSTM model (red). During rollout, the model collapses after around 500 steps.

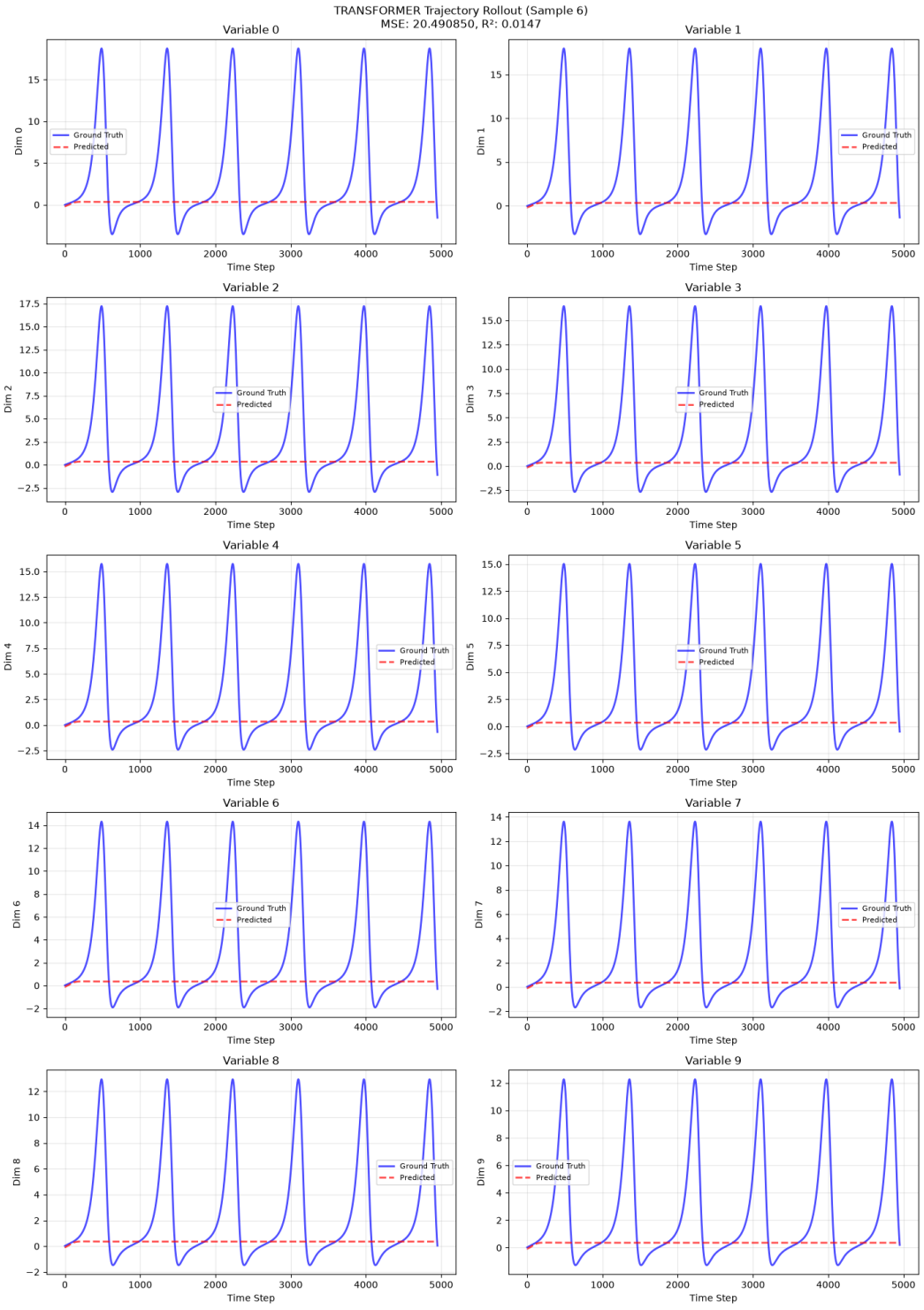
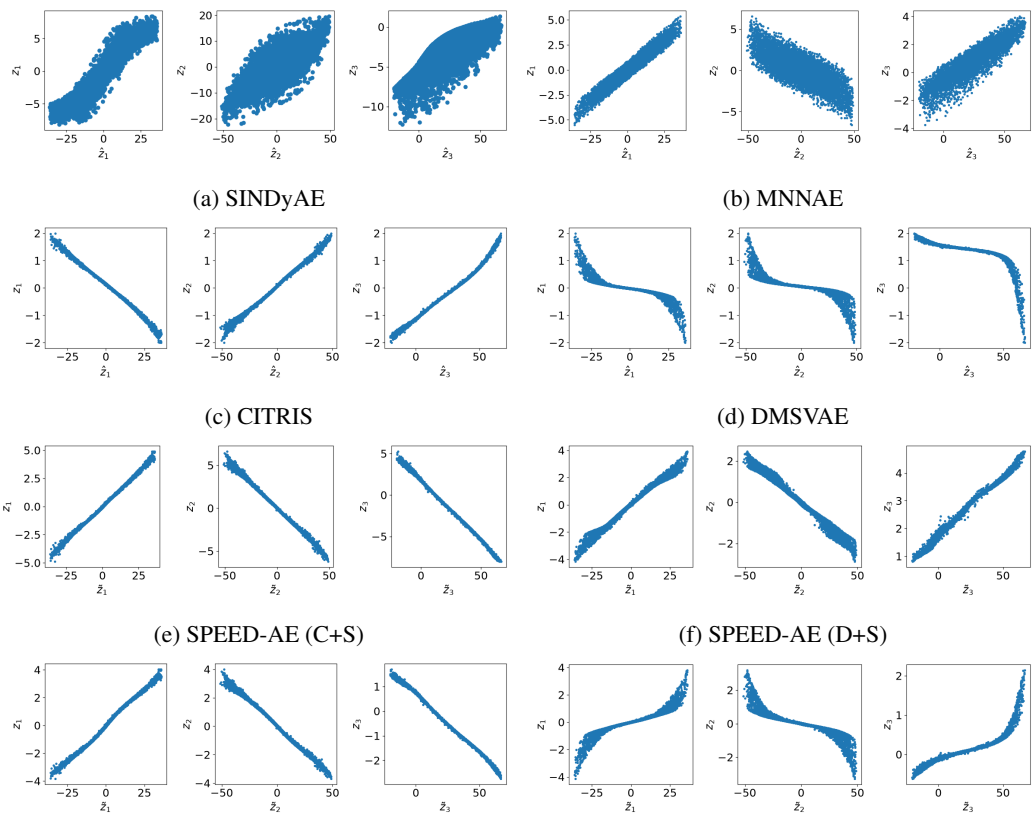


Figure 14: Example test trajectory of the Lotka-Volterra experiment (first 10 dimensions of $\mathbf{x}$). Ground truth (blue) against the predicted trajectory from the Transformer model (red). During rollout, the model collapses after a few iterations.



(a) SINDyAE (b) MNNAE

(c) CITRIS (d) DMSVAE

(e) SPEED-AE (C+S) (f) SPEED-AE (D+S)

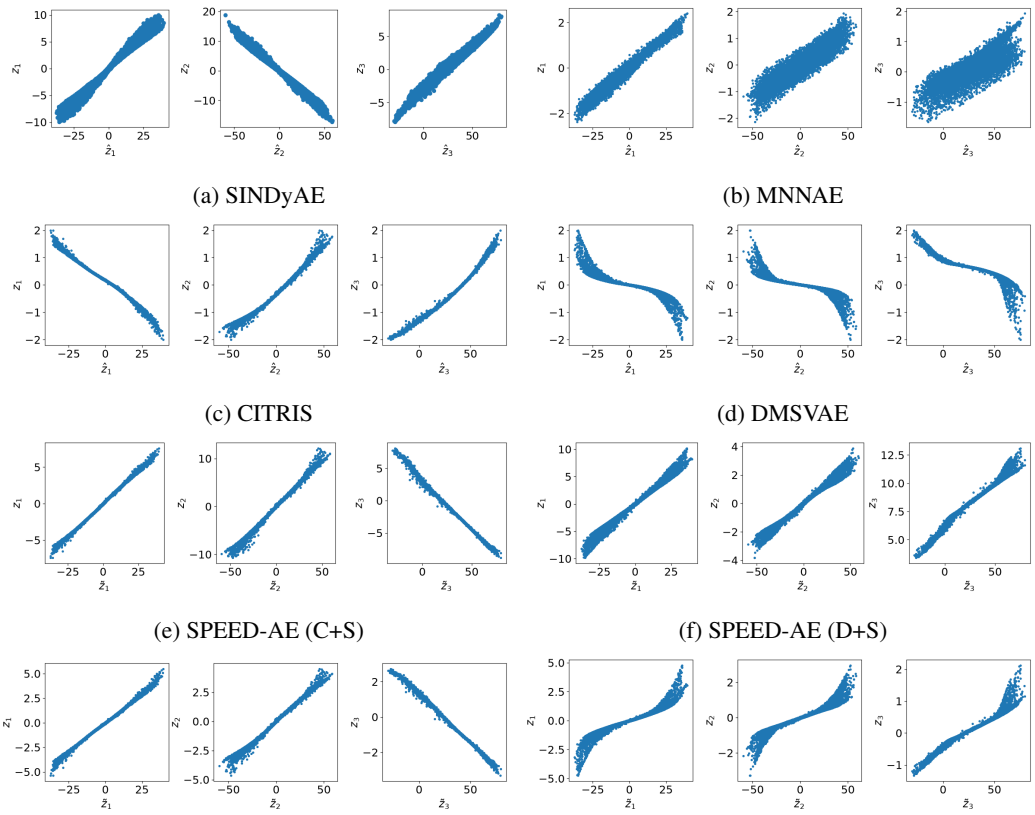
(g) SPEED-AE (C+M) (h) SPEED-AE (D+M)

Figure 15: Lorenz experiment with stable trajectories $\rho = 14$. Scatter plots of the found latents $\tilde{\mathbf{z}}$ against the true ones $\mathbf{z}$. If their function is well defined, the plots well represent the maps $z_i = h_i(\tilde{z}_i)$. In the second row, we include the same plots for the CRL latents $\hat{\mathbf{z}}$ from CITRIS and DMSVAE. SPEED-AE models with DMSVAE as disentangler suffer more due to the CRL latents being less disentangled, although the SINDy-based loss is actually able to recover a linear map h.

H.2 Lorenz Experiment

We report similar scatter plots for the Lorenz stable and chaotic experiments, respectively, in Figure 15 and Figure 16, with cross-correlation matrices in Figure 9c and Figure 9d. SINDyAE learns variables with good disentanglement and cross-correlation in the chaotic case, which data and parameters come directly from the original work of Champion et al. (2019), but struggles in the stable case. MNNAE has strong forecasting performance, but learns variables that do not easily relate to the true ones. In these experiments, DMSVAE performs significantly worse than CITRIS. However, SPEED-AE is actually able to improve both in terms of cross-correlation and identifiability (compare Figure 15d with Figure 15f and Figure 16d with Figure 16f, and the matrices in Figure 9c).

Finally, we report the identified ODE coefficients in Figure 17.



(a) SINDyAE (b) MNNAE

(c) CITRIS (d) DMSVAE

(e) SPEED-AE (C+S) (f) SPEED-AE (D+S)

(g) SPEED-AE (C+M) (h) SPEED-AE (D+M)

Figure 16: Scatter plots of the found latents $\tilde{\mathbf{z}}$ against the true ones $\mathbf{z}$. If their function is well-defined, the plots well represent the maps $z_i = h_i(\tilde{z}_i)$. In the second row, we include the same plots for the CRL latents $\hat{\mathbf{z}}$ from CITRIS and DMSVAE. SPEED-AE models with DMSVAE as disentangler suffer more due to the CRL latents being less disentangled, although the SINDy-based loss is actually able to recover a linear map h.

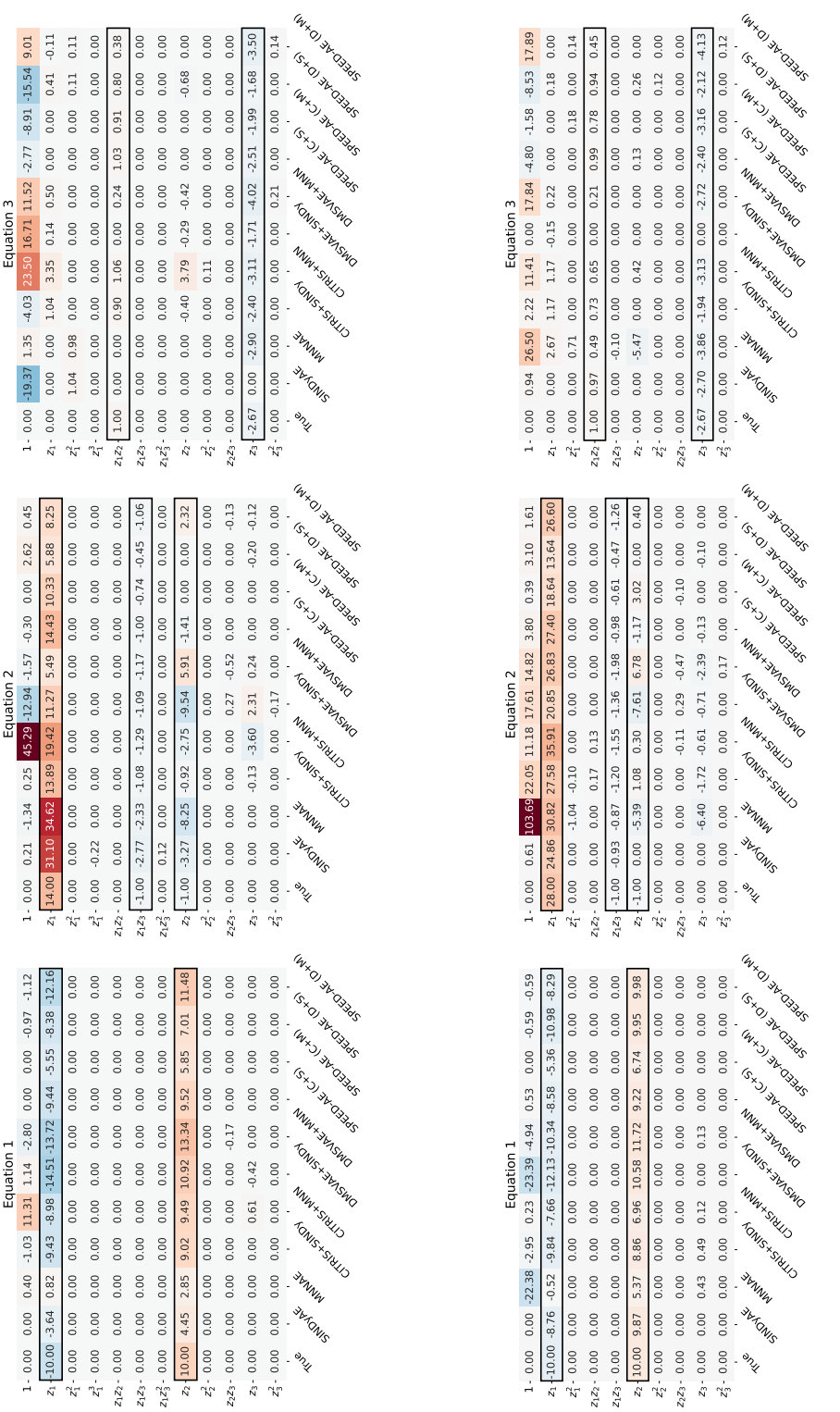


Figure 17: Identified coefficients for the Lorenz experiments in the stable ($\rho = 14$, left) and chaotic case ($\rho = 28$, right). The true ones are in the leftmost column and highlighted by the black borders. For comparison, we include four baselines that consist of equation discovery directly applied to the CRL latents $\hat{\mathbf{z}}$.

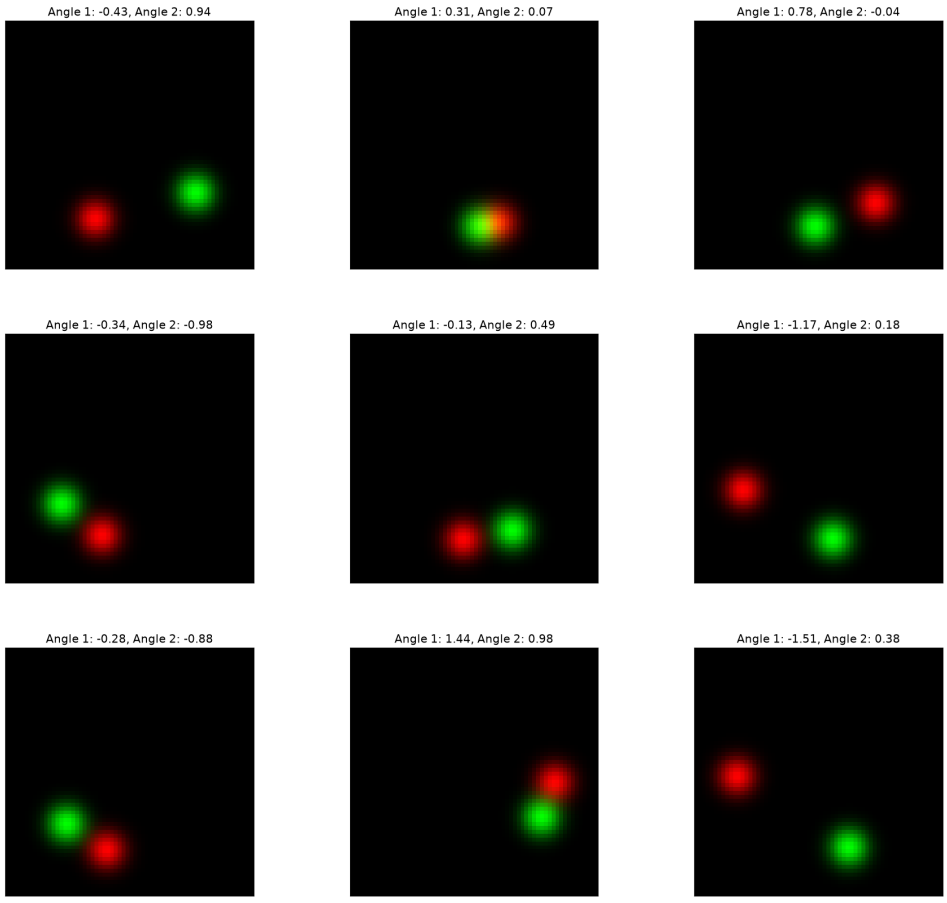
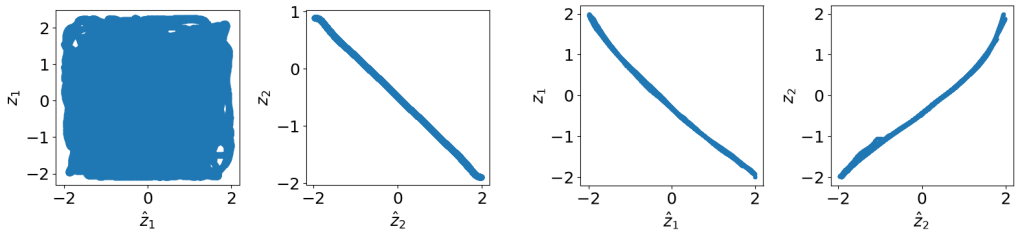


Figure 18: Example images from the experiment with two pendulums. Pendulums are colored differently to ensure invertibility of the mixing function.

H.3 Pendulums Experiment

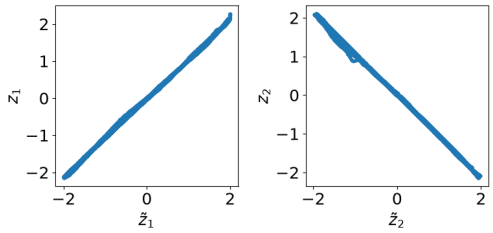
We first include example images of the dataset in Figure 18. Different pendulums are represented on different channels to ensure the mixing function is invertible and the two pendulums are distinguishable. This should also help disentanglement for all models.

Figure 19 shows the scatter plots of true against found latents. CITRIS achieves disentanglement, but not identifiability up to a linear function, where SPEED-AE succeeds. SINDyAE has a very interesting behavior: it learns a variable that is linearly related to one of the two pendulum angles, while the other is completely disaligned. Since the pendulums have equal dynamics, and thus their motion is the same except for a constant phase, SINDyAE might have learned one variable for the common dynamics and one that maps the two pendulums to each other. This, however, has the limitation of not being able to identify one of the two pendulums correctly. We identify this behavior in all 10 seeds we ran the experiment with.



(a) SINDyAE

(b) CITRIS



(c) SPEED-AE (C+S)

Figure 19: Pendulums experiment. Scatter plots of the found latents $\tilde{\mathbf{z}}$ against the true ones $\mathbf{z}$. If their function is well defined, the plots well represent the maps $z_i = h_i(\tilde{z}_i)$. In the second plot, we include the same for the CRL latents $\hat{\mathbf{z}}$ from CITRIS. SPEED-AE is able to achieve linear identifiability from the general diffeomorphism of CITRIS. SINDyAE only identifies one variable, while the other likely represents a mapping between the two pendulums.