

# <sup>1</sup> GD-VAEs Package: Geometric Dynamic Variational Autoencoders

<sup>3</sup> Ryan Lopez<sup>1</sup> and Paul J. Atzberger  <sup>1</sup>

<sup>4</sup> 1 University California Santa Barbara

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- <sup>5</sup> [Review](#) 
- <sup>6</sup> [Repository](#) 
- <sup>7</sup> [Archive](#) 

Editor: [Open Journals](#) 

## Reviewers:

- <sup>8</sup> [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright  
<sup>17</sup> and release the work under a  
<sup>18</sup> Creative Commons Attribution 4.0  
International License ([CC BY 4.0](#)).  
<sup>19</sup>

## Summary

<sup>20</sup> The GD-VAEs software package provides approaches for data-driven learning of representations  
<sup>21</sup> for states and non-linear dynamics over both standard and general manifold latent spaces.  
<sup>22</sup> Methods are provided allowing for representations incorporating information in the form of  
<sup>23</sup> geometric and topological structures arising from constraints, periodicity, and other properties  
<sup>24</sup> of the dynamics. Training approaches are provided for learning encoders and decoders using  
<sup>25</sup> training strategies related to Variational Autoencoders (VAEs). Representations can be  
<sup>26</sup> learned based on deep neural network architectures that include general Multilayer Perceptrons  
<sup>27</sup> (MLPs), Convolutional Neural Networks (CNNs), and Transpose CNNs (T-CNNs). Motivating  
<sup>28</sup> applications include parameterized PDEs, constrained mechanical systems, reductions in non-  
<sup>29</sup> linear systems, and other tasks involving dynamics. The package is implemented currently  
<sup>30</sup> in PyTorch. The source code for this initial version 1.0.0 of GD-VAEs has been archived to  
<sup>31</sup> Zenodo with a DOI in ([Atzberger, 2023](#)). For related papers, examples, updates, and additional  
<sup>32</sup> information see <https://github.com/gd-vae/gd-vae> and <http://atzberger.org/>.  
<sup>33</sup>

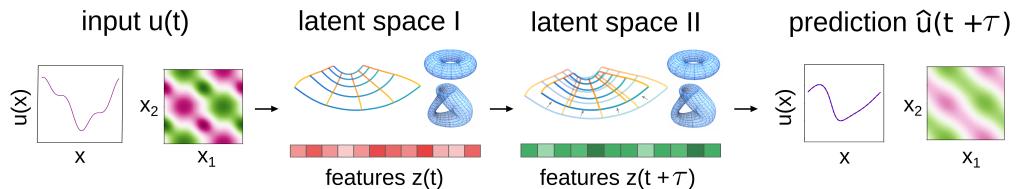
## Statement of Need

<sup>20</sup> A central challenge in learning non-linear dynamics is to obtain representations not only capable  
<sup>21</sup> of reproducing similar outputs as observed in the training data set but to infer more inherent  
<sup>22</sup> structures for performing simulations or making long-time predictions. We develop methods for  
<sup>23</sup> learning more robust non-linear models by providing ways to incorporate underlying structural  
<sup>24</sup> information related to smoothness, physical principles, topology/geometry, and other properties  
<sup>25</sup> of the dynamics. We focus particularly on developing generative models using Probabilistic  
<sup>26</sup> Autoencoders (PAEs) that incorporate noise-based regularizations and priors on manifold latent  
<sup>27</sup> spaces to learn lower dimensional representations from observations. This provides the basis  
<sup>28</sup> of non-linear state space models for simulations and predictions. The methods provide ways  
<sup>29</sup> to learning representations over both standard and more general manifold latent spaces with  
<sup>30</sup> prescribed topology/geometry. This facilitates capturing both quantitative and qualitative  
<sup>31</sup> features of the dynamics to enhance robustness and interpretability of results. The methods are  
<sup>32</sup> motivated by applications including parameterized PDEs, reduced order modeling, constrained  
<sup>33</sup> mechanical systems, and other tasks involving dynamics.

## Data-Driven Modeling of Dynamics

<sup>35</sup> Many problem domains and tasks require learning from observations a set of models for  
<sup>36</sup> dynamics. The most common approach is to develop approximations based on linear dynamical  
<sup>37</sup> systems (LDS). These include the Kalman Filter and extensions ([Del Moral, 1997; Godsill,](#)  
<sup>38</sup> [Kalman, 1960; Van Der Merwe et al., 2000; Wan & Van Der Merwe, 2000](#)), Proper  
<sup>39</sup> Orthogonal Decomposition (POD) ([Chatterjee, 2000; Mendez et al., 2018](#)), and more recently

40 Dynamic Mode Decomposition (DMD) (Kutz et al., 2016; Schmid, 2010; Tu et al., 2014) and  
 41 Koopman Operator approaches (Das & Giannakis, 2019; Korda et al., 2020; Mezić, 2013).  
 42 These methods utilize the linearity allowing for strong assumptions about the model structure  
 43 to be utilized to develop effective algorithms.  
 44 Obtaining representations for more general non-linear dynamics poses challenges and less  
 45 unified approaches given the wide variety of possible system behaviors. For classes of systems  
 46 and specific application domains, methods have been developed which make different levels of  
 47 assumptions about the underlying structure of the dynamics. Methods for learning non-linear  
 48 dynamics include the NARX and NOE approaches with function approximators based on  
 49 neural networks and other models classes (Nelles, 2013; Sjöberg et al., 1995), sparse symbolic  
 50 dictionary methods that are linear-in-parameters such as SINDy (Kutz et al., 2016; Schmidt &  
 51 Lipson, 2009; Sjöberg et al., 1995), and dynamic Bayesian networks (DBNs), such as Hidden  
 52 Markov Chains (HMMs) and Hidden-Physics Models (Baum & Petrie, 1966; Ghahramani &  
 53 Roweis, 1998; Krishnan et al., 2017; Pawar et al., 2020; Raissi & Karniadakis, 2018; Saul,  
 54 2020). Related research has also been done in non-linear system identification, including  
 55 (Archer et al., 2015; Chiuso & Pillonetto, 2019; Schoukens & Ljung, 2019). The strategies  
 56 and methods often overlap between fields, but with different terminology depending on the  
 57 particular research community and applications.



**Figure 1:** Data-driven modeling of non-linear dynamics for simulations and prediction. From observation data, representations are learned within latent spaces by training encoders and decoders to preserve relevant information. The GD-VAEs package provides methods for data-driven modeling of non-linear dynamics using a Variational Autoencoder (VAE) framework with both standard and general manifold latent spaces. Methods are provided for incorporating geometric and topological information into the latent space representations.

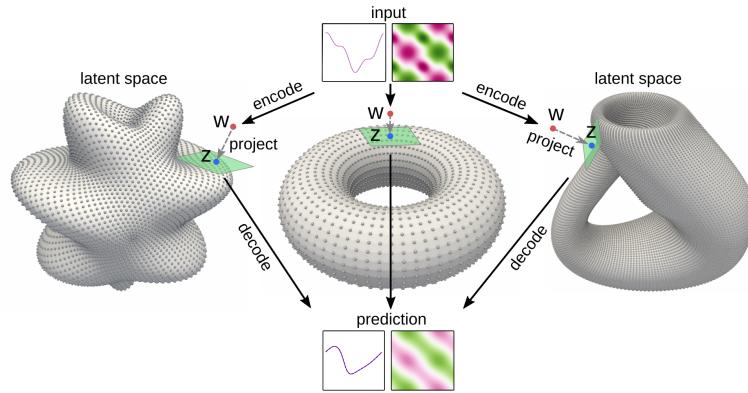
58 For many systems, parsimonious representations can be obtained by working with non-euclidean  
 59 manifold latent spaces, such as a torus for doubly periodic systems or even non-orientable  
 60 manifolds, such as a klein bottle as arises in imaging and perception studies (Carlsson et  
 61 al., 2008). For this purpose, we learn encoders  $\mathcal{E}$  over a family of mappings to a prescribed  
 62 manifold  $\mathcal{M}$  of the form

$$z = \mathcal{E}_\phi(x) = \Lambda(\tilde{\mathcal{E}}_\phi(x)) = \Lambda(w), \text{ where } w = \tilde{\mathcal{E}}_\phi(x).$$

63 The  $\mathcal{E}_\phi$  is a candidate encoder to the manifold with the parameters  $\phi$ .

64 To generate a family of maps over which we can learn in practice, we use that a smooth  
 65 closed manifold  $\mathcal{M}$  of dimension  $m$  can be embedded within  $\mathbb{R}^{2m}$ , as supported by the  
 66 Whitney Embedding Theorem (Whitney, 1944). We obtain a family of maps to the manifold  
 67 by constructing maps in two steps using the expressions above. In the first step, we use an  
 68 unconstrained encoder  $\tilde{\mathcal{E}}$  from  $x$  to a point  $w$  in the embedding space. In the second step, we  
 69 use a map  $\Lambda$  that projects a point  $w \in \mathbb{R}^{2m}$  to a point  $z \in \mathcal{M} \subset \mathbb{R}^{2m}$  within the embedded  
 70 manifold. In this way,  $\tilde{\mathcal{E}}$  can be any learnable mapping from  $\mathbb{R}^n$  to  $\mathbb{R}^{2m}$ , for which there  
 71 are many model classes including neural networks. To obtain a particular manifold map, the  
 72  $\tilde{\mathcal{E}}$  only needs to learn an equivalent mapping from  $x$  to  $w$ , where  $w$  is in the appropriate  
 73 equivalence class  $\mathcal{Q}_z$  of a target point  $z$  on the manifold,  $w \in \mathcal{Q}_z = \{w \mid \Lambda(w) = z\}$ . Here,  
 74 we accomplish this in practice two ways: (i) we provide an analytic mapping  $\Lambda$  to  $\mathcal{M}$ , (ii) we  
 75 provide a high resolution point-cloud representation of the target manifold along with local

<sup>76</sup> gradients and use for  $\Lambda$  a quantized or interpolated mapping to the nearest point on  $\mathcal{M}$ . More  
<sup>77</sup> details can be found in (Lopez & Atzberger, 2022).



**Figure 2:** Manifold Latent Spaces and Learnable Mappings. Mappings are developed for using latent space representations having general geometries and topologies. Differentiable model classes are used amenable to backpropagation, such as neural networks, for incorporating into machine learning training frameworks. This is done by mapping inputs first to a point in the embedding space which is then projected to a point in the manifold. The maps can handle the manifold and compute projections based on general point cloud representations, analytic descriptions, product spaces, or other descriptions.

<sup>78</sup> In practice, we can view the projection map  $z = \Lambda(w)$  to the manifold as the solution of the  
<sup>79</sup> optimization problem

$$z^* = \arg \min_{z \in \mathcal{M}} \frac{1}{2} \|w - z\|_2^2.$$

<sup>80</sup> We can always express patches of a smooth manifold using local coordinate charts  $z = \sigma^k(u)$   
<sup>81</sup> for  $u \in \mathcal{U} \subset \mathbb{R}^m$ . For example, we could use in practice a local Monge-Gauge quadratic fit  
<sup>82</sup> to a point cloud representation of the manifold, as in (Gross et al., 2020). We can express  
<sup>83</sup>  $z^* = \sigma^{k^*}(u^*)$  for some chart  $k^*$  for solution of the optimization problem. In terms of the  
<sup>84</sup> collection of coordinate charts  $\{\mathcal{U}^k\}$  and local parameterizations  $\{\sigma^k(u)\}$ , we can express this  
<sup>85</sup> as

$$u^*, k^* = \arg \min_{k, u \in \mathcal{U}^k} \Phi_k(u, w), \text{ where } \Phi_k(u, w) = \frac{1}{2} \|w - \sigma^k(u)\|_2^2.$$

<sup>86</sup> The  $w$  is the input and  $u^*, k^*$  is the solution. This gives the coordinate-based representation  
<sup>87</sup>  $z^* = \sigma^{k^*}(u^*) = \Lambda(w)$ . For smooth parameterizations  $\sigma(u)$ , the optimal solutions  $u^*(w)$   
<sup>88</sup> satisfies from the optimization procedure the following implicit equation

$$G(u^*, w) := \nabla_u \Phi_{k^*}(u^*, w) = 0.$$

<sup>89</sup> During learning with backpropagation, we need to be able to compute the gradient

$$\nabla_\phi z^* = \nabla_\phi \sigma^k(u^*) = \nabla_\phi \Lambda(\tilde{\mathcal{E}}_\phi(x)) = \nabla_w \Lambda(w) \nabla_\phi \tilde{\mathcal{E}}_\phi,$$

<sup>90</sup> where  $w = \tilde{\mathcal{E}}_\phi$ . If we approach training models using directly these expressions, we would need  
<sup>91</sup> ways to compute both the gradients  $\nabla_\phi \tilde{\mathcal{E}}_\phi$  and  $\nabla_w \Lambda(w)$ . While the gradients  $\nabla_\phi \tilde{\mathcal{E}}_\phi$  can  
<sup>92</sup> be obtained readily for many model classes, such as neural networks using backpropagation,  
<sup>93</sup> the gradients  $\nabla_w \Lambda(w)$  pose additional challenges. If  $\Lambda$  can be expressed analytically then  
<sup>94</sup> backpropagation techniques in principle may still be employed directly. However, in practice  $\Lambda$   
<sup>95</sup> will often result from a numerical solution of the optimization problem. We show how in this  
<sup>96</sup> setting alternative approaches can be used to obtain the gradient  $\nabla_\phi z^* = \nabla_\phi \Lambda(\tilde{\mathcal{E}}_\phi(x))$ .

<sup>97</sup> To obtain gradients  $\nabla_\phi z^*$ , we derive expressions by considering variations  $w = w(\gamma)$ ,  $\phi = \phi(\gamma)$   
<sup>98</sup> for a scalar parameter  $\gamma$ . For example, this can be motivated by taking  $w(\gamma) = \tilde{\mathcal{E}}_\phi(x(\gamma))$  and

99      $\phi = \phi(\gamma)$  for some path  $(\mathbf{x}(\gamma), \phi(\gamma))$  in the input and parameter space  $(\mathbf{x}, \phi) \in \mathcal{X} \times \mathcal{P}$ . We  
 100    can obtain the needed gradients by determining the variations of  $\mathbf{u}^* = \mathbf{u}^*(\gamma)$ . This follows  
 101    since  $\mathbf{z}^* = \sigma^k(\mathbf{u}^*)$  and  $\nabla_\phi \mathbf{z}^* = \nabla_{\mathbf{u}} \sigma^k(\mathbf{u}^*) \nabla_\phi \mathbf{u}^*$ . The  $\nabla_{\mathbf{u}} \sigma^k(\mathbf{u}^*)$  often can be readily obtained  
 102    numerically or from backpropagation. This allows us to express the gradients using the Implicit  
 103    Function Theorem as

$$0 = \frac{d}{d\gamma} G(\mathbf{u}^*(\gamma), \mathbf{w}(\gamma)) = \nabla_{\mathbf{u}} G \frac{d\mathbf{u}^*}{d\gamma} + \nabla_{\mathbf{w}} G \frac{d\mathbf{w}}{d\gamma}.$$

104    The term typically posing the most significant computational challenge is  $d\mathbf{u}^*/d\gamma$  since  $\mathbf{u}^*$  is  
 105    obtained numerically from the optimization problem. We solve for it using the expressions to  
 106    obtain

$$\frac{d\mathbf{u}^*}{d\gamma} = -[\nabla_{\mathbf{u}} G]^{-1} \nabla_{\mathbf{w}} G \frac{d\mathbf{w}}{d\gamma}.$$

107    This only requires that we can evaluate for a given  $(\mathbf{u}, \mathbf{w})$  the local gradients  $\nabla_{\mathbf{u}} G$ ,  $\nabla_{\mathbf{w}} G$ ,  
 108     $d\mathbf{w}/d\gamma$ , and use that  $\nabla_{\mathbf{u}} G$  is invertible. Computationally, this only requires us to find  
 109    numerically the solution  $\mathbf{u}^*$  and evaluate numerically the expression for a given  $(\mathbf{u}^*, \mathbf{w})$ . This  
 110    allows us to avoid needing to compute directly  $\nabla_{\mathbf{w}} \Lambda_{\mathbf{w}}(\mathbf{w})$ . This provides an alternative  
 111    practical approach for computing  $\nabla_\phi \mathbf{z}^*$  useful in training models.

112    For learning via backpropagation, we use these results to assemble the needed gradients for our  
 113    manifold encoder maps  $\mathcal{E}_\theta = \Lambda(\tilde{\mathcal{E}}_\theta(\mathbf{x}))$  as follows. Using  $\mathbf{w} = \tilde{\mathcal{E}}_\theta(\mathbf{x})$ , we first find numerically  
 114    the closest point in the manifold  $\mathbf{z}^* \in \mathcal{M}$  and represent it as  $\mathbf{z}^* = \sigma(\mathbf{u}^*) = \sigma^{k^*}(\mathbf{u}^*)$  for some  
 115    chart  $k^*$ . Next, using this chart we compute the gradients using that

$$G = \nabla_{\mathbf{u}} \Phi(\mathbf{u}, \mathbf{w}) = -(\mathbf{w} - \sigma(\mathbf{u}))^T \nabla_{\mathbf{u}} \sigma(\mathbf{u}).$$

116    We use a column vector convention with  $\nabla_{\mathbf{u}} \sigma(\mathbf{u}) = [\sigma_{u_1} | \dots | \sigma_{u_k}]$ . We next compute

$$\nabla_{\mathbf{u}} G = \nabla_{\mathbf{u}\mathbf{u}} \Phi = \nabla_{\mathbf{u}} \sigma^T \nabla_{\mathbf{u}} \sigma - (\mathbf{w} - \sigma(\mathbf{u}))^T \nabla_{\mathbf{u}\mathbf{u}} \sigma(\mathbf{u})$$

117    and

$$\nabla_{\mathbf{w}} G = \nabla_{\mathbf{w}, \mathbf{u}} \Phi = -I \nabla_{\mathbf{u}} \sigma(\mathbf{u}).$$

118    From the gradients  $\nabla_{\mathbf{u}} G$ ,  $\nabla_{\mathbf{w}} G$ , we compute  $\nabla_\phi \mathbf{z}^*$ . This allows us to learn VAEs with latent  
 119    spaces for  $\mathbf{z}$  with general specified topologies and controllable geometric structures. For more  
 120    details see ([Lopez & Atzberger, 2022](#)).

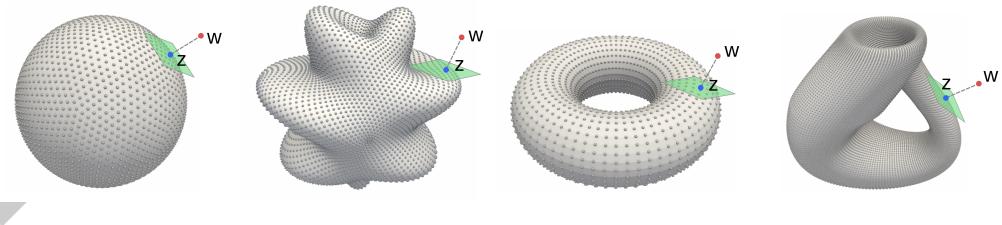


Figure 3: Manifold Latent Spaces

121    We develop data-driven modeling approaches based on a Variational Autoencoder (VAE)  
 122    framework ([Kingma & Welling, 2014](#)). From observation data of the dynamics, we develop  
 123    learning methods for obtaining representations within latent spaces for performing simulations  
 124    or for making long-time predictions. In practice, data can include experimental measurements,  
 125    large-scale computational simulations, or solutions of complicated dynamical systems for which  
 126    we seek reduced models. Representations and reductions can aid in gaining insights for a class  
 127    of inputs or for physical regimes to understand better underlying mechanisms generating the  
 128    observed behaviors. Such representations are also helpful for performing optimization and in  
 129    the development of controllers ([Nelles, 2013](#)).

130 Standard autoencoders can result in encodings of observations  $\mathbf{x}$  that yield unstructured scat-  
 131 tered disconnected coding points  $\mathbf{z}$  representing the system features. VAEs provide probabilistic  
 132 encoders and decoders where noise provides regularizations that promote more connected en-  
 133 codings, smoother dependence on inputs, and more disentangled feature components ([Kingma & Welling, 2014](#)). In addition, we also use and provide in the package methods for other  
 134 regularizations to help aid with interpretability and enhance stability of the learned latent  
 135 representations.  
 136

## 137 Variational Autoencoder (VAE) Framework for Dynamics

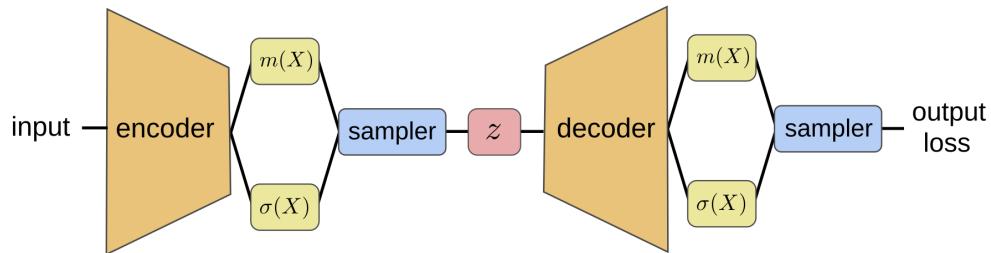
138 We use a Maximum Likelihood Estimation (MLE) approach based on the Log Likelihood (LL)  
 139  $\mathcal{L}_{LL} = \log(p_\theta(\mathbf{X}, \mathbf{x}))$  to learn VAE predictors. We consider the dynamics of  $u(s)$  and let  
 140  $\mathbf{X} = u(t)$  and  $\mathbf{x} = u(t + \tau)$ . The  $p_\theta$  is based on the generative model of the autoencoder  
 141 framework shown in the figures. We use variational inference to approximate the LL by the  
 142 Evidence Lower Bound (ELBO) ([Blei et al., 2017](#)). We train a model with parameters  $\theta$  using  
 143 encoders and decoders based on minimizing the loss function

$$144 \theta^* = \arg \min_{\theta_e, \theta_d} -\mathcal{L}^B(\theta_e, \theta_d, \theta_\ell; \mathbf{X}^{(i)}, \mathbf{x}^{(i)}), \quad \mathcal{L}^B = \mathcal{L}_{RE} + \mathcal{L}_{KL} + \mathcal{L}_{RR},$$

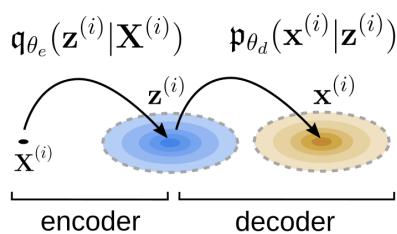
$$145 \mathcal{L}_{RE} = E_{q_{\theta_e}(\mathbf{z}|\mathbf{X}^{(i)})} [\log p_{\theta_d}(\mathbf{x}^{(i)}|\mathbf{z}')], \quad \mathcal{L}_{KL} = -\beta \mathcal{D}_{KL}(q_{\theta_e}(\mathbf{z}|\mathbf{X}^{(i)}) \| \tilde{p}_{\theta_d}(\mathbf{z})), \\ \mathcal{L}_{RR} = \gamma E_{q_{\theta_e}(\mathbf{z}'|\mathbf{x}^{(i)})} [\log p_{\theta_d}(\mathbf{x}^{(i)}|\mathbf{z}')].$$

146 The  $q_{\theta_e}$  denotes the encoding probability distribution and  $p_{\theta_d}$  the decoding probability distribu-  
 147 tion. The loss  $\ell = -\mathcal{L}^B$  provides a regularized form of MLE. The term  $\mathcal{L}^B$  approximates  
 148 the likelihood of the encoder-decoder generative model fitting the training data. This can  
 149 be decomposed into the following three terms (i)  $\mathcal{L}_{RR}$  is the log likelihood of reconstructing  
 150 samples, (ii)  $\mathcal{L}_{RE}$  is the log likelihood of predicting samples after a single time step, and (iii)  
 151  $\mathcal{L}_{KL}$  is a regularization term associated with a prior distribution on the latent space.

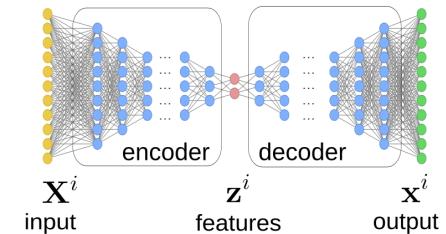
## Variational Autoencoders (VAEs)



## VAE Probabilistic Mappings



## Deep Neural Network



**Figure 4:** Dynamic Variational Autoencoder (D-VAE) Framework. The encoders and decoders are trained based on variational approximation of a generative autoencoder model (Kingma & Welling, 2014). We develop related approaches to learn representations of the non-linear dynamics. Deep Neural Networks (DNNs) are trained (i) to serve as feature extractors to represent functions  $\mathbf{X}^{(i)} = u(\tilde{x}, t)$ , and their evolution, in a low dimensional latent space as  $\mathbf{z}(t)$  (probabilistic encoder  $\sim q_{\theta_e}$ ), and (ii) to serve as approximators that can construct predictions  $\mathbf{x}^{(i)} = u(\tilde{x}, t + \tau)$  using features  $\mathbf{z}(t + \tau)$  (probabilistic decoder  $\sim p_{\theta_d}$ ).

152 The terms  $\mathcal{L}_{RE}$  and  $\mathcal{L}_{KL}$  arise from the ELBO variational bound  $\mathcal{L}_{LL} \geq \mathcal{L}_{RE} + \mathcal{L}_{KL}$   
 153 when  $\beta = 1$ , (Blei et al., 2017). This provides a way to estimate the log likelihood that the  
 154 encoder-decoder reproduce the observed data sample pairs  $(\mathbf{X}^{(i)}, \mathbf{x}^{(i)})$  using the codes  $\mathbf{z}'$  and  
 155  $\mathbf{z}$ . Here, we include a latent-space mapping  $\mathbf{z}' = f_{\theta_\ell}(\mathbf{z})$  which is parameterized by  $\theta_\ell$ . The  
 156 mapping can be prescribed or learned during training over a class of maps. The latent-space  
 157 mapping can be used to characterize the evolution of the system or for further processing of  
 158 features. The  $\mathbf{X}^{(i)}$  is the input and  $\mathbf{x}^{(i)}$  is the output prediction. For the case of dynamical  
 159 systems, we take  $\mathbf{X}^{(i)} \sim u^i(t)$  a sample of the initial state function  $u^i(t)$  and the output  
 160  $\mathbf{x}^{(i)} \sim u^i(t + \tau)$  the predicted state function  $u^i(t + \tau)$ . We discuss the specific distributions  
 161 used in more detail below.

162 The  $\mathcal{L}_{KL}$  term involves the Kullback-Leibler Divergence (Cover & Thomas, 2006; Kullback  
 163 & Leibler, 1951) acting similar to a Bayesian prior on latent space to regularize the encoder  
 164 conditional probability distribution  $p_{\theta}(\mathbf{z}|\mathbf{X})$  so that for each sample this distribution is similar  
 165 to  $p_{\theta_d}$ . We take  $p_{\theta_d} = \eta(0, \sigma_0^2)$  a multi-variate Gaussian with independent components. This  
 166 serves (i) to disentangle the features from each other to promote independence, (ii) provide  
 167 a reference scale and localization for the encodings  $\mathbf{z}$ , and (iii) promote parsimonious codes  
 168 utilizing smaller dimensions than  $d$  when possible. The  $\mathcal{L}_{RR}$  term gives a regularization that  
 169 promotes retaining information in  $\mathbf{z}$  so the encoder-decoder pair can reconstruct functions.  
 170 This also promotes organization of the latent space for consistency over multi-step predictions  
 171 and aids in the model interpretability.

172 We use for the specific encoder probability distributions conditional Gaussians  $\mathbf{z} \sim q_{\theta_e}(\mathbf{z}|\mathbf{x}^{(i)}) =$   
 173  $\alpha(\mathbf{X}^{(i)}, \mathbf{x}^{(i)}) + \eta(0, \sigma_e^2)$  where  $\eta$  is a Gaussian with variance  $\sigma_e^2$ , (i.e.  $\mathbb{E}^{\mathbf{X}^i}[\mathbf{z}] = \alpha$ ,  $\text{Var}^{\mathbf{X}^i}[\mathbf{z}] =$   
 174  $\sigma_e^2$ ). One can think of the learnable mean function  $\alpha$  in the VAE as corresponding to a typical  
 175 encoder  $\alpha(\mathbf{X}^{(i)}, \mathbf{x}^{(i)}; \theta_e) = \alpha(\mathbf{X}^{(i)}; \theta_e) = \mathbf{z}^{(i)}$  and the variance function  $\sigma_e^2 = \sigma_e^2(\theta_e)$  as

176 providing control of a noise source to further regularize the encoding. The  $\alpha$  can be represented  
 177 by a deep neural network or other model classes. Among other properties, using noise in the  
 178 encoding promotes connectedness of the ensemble of latent space codes and smoothness, since  
 179 encoders and decoders need to produce similar responses for nearby codes.

180 For the VAE decoder distribution, we take  $\mathbf{x} \sim p_{\theta_d}(\mathbf{x}|\mathbf{z}^{(i)}) = b(\mathbf{z}^{(i)}) + \eta(0, \sigma_d^2)$ . The  
 181 learnable mean function  $b(\mathbf{z}^{(i)}; \theta_e)$  corresponds to a typical deterministic decoder and the  
 182 variance function  $\sigma_e^2 = \sigma_e^2(\theta_d)$  controls the source of regularizing noise. In practice, while the  
 183 variances are learnable for many problems it can be useful to treat the  $\sigma(\cdot)$  as hyper-parameters.  
 184 We discuss in the code usage and examples for how to adjust these encoder-decoder models  
 185 and the VAE training in the GD-VAE package.

186 As a summary, the key terms to be learned in the GD-VAE dynamical models are  $(\alpha, \sigma_e, f_{\theta_e}, b, \sigma_d)$   
 187 which are parameterized by  $\theta = (\theta_e, \theta_d, \theta_\ell)$ . We learn predictors for the dynamics by training  
 188 over samples of evolution pairs  $\{(u_n^i, u_{n+1}^i)\}_{i=1}^m$ , where  $i$  denotes the sample index and  
 189  $u_n^i = u^i(t_n)$  with  $t_n = t_0 + n\tau$  for a time-scale  $\tau$ . To make predictions, the learned models  
 190 use the following stages: (i) extract from  $u(t)$  the features  $z(t)$ , (ii) evolve  $z(t) \rightarrow z(t + \tau)$ ,  
 191 (iii) predict using  $z(t + \tau)$  the  $\hat{u}(t + \tau)$ . By composition of the latent evolution maps the  
 192 models can make multi-step predictions of the dynamics. For additional discussions see ([Lopez & Atzberger, 2022](#)).  
 193

## 194 Package Organization

195 The current implementation of the GD-VAEs package is organized into modules for handling  
 196 the mappings to the manifold latent spaces, learning of dynamic variational autoencoders,  
 197 and using different model classes and neural network architectures for the encoders, decoders,  
 198 and dynamic maps. The current codes are currently implemented within the machine learning  
 199 framework PyTorch and use a modular interface for running each of the example cases drawing  
 200 on the GD-VAE methods.

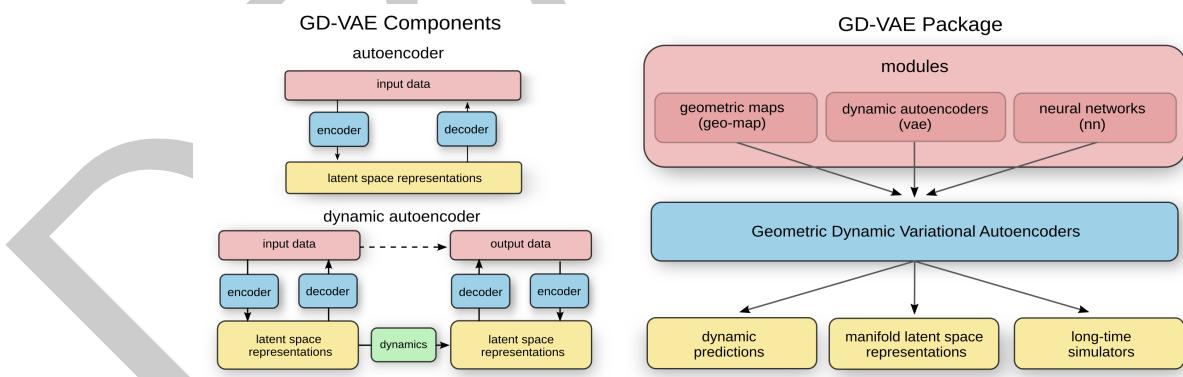


Figure 5: The GD-VAEs package is organized for handling different types of autoencoders for learning general representations and for learning dynamics. The package is organized into modules for handling the manifold latent spaces, dynamic variational autoencoders, and neural network architectures including Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs).

201 The geo-map module allows for incorporating topological and geometric information into the  
 202 learned representations. Methods are implemented both for explicit analytic maps to common  
 203 topological spaces and for handling general point cloud representations of the geometry. This  
 204 allows for flexibility in how the manifold latent spaces are specified, without the need for  
 205 explicit analytic expressions. The point cloud representations provide a way to represent the  
 206 manifold in terms of an embedding within  $\mathbb{R}^d$ . In this way, latent spaces with general topology

207 can be used, including non-orientable manifolds. This is illustrated in an example case using a  
 208 Klein Bottle manifold.

209 The dynamic VAE methods and related formulations are implemented in our vae module,  
 210 which handles using ELBO variational approximations for autoencoder generative models. This  
 211 includes implementations of both prediction and reconstruction loss terms and regularization  
 212 terms. This also includes the samplers for the reparameterization estimators for the ELBO for  
 213 use in the stochastic gradient descent optimization.

214 The deep learning neural network architectures for use in the autoencoders and decoders are  
 215 implemented in our nn module. This includes implementations for encoders/decoders based  
 216 on Multi-layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Transpose  
 217 Convolutional Neural Networks (T-CNNs). This also includes methods interfacing with the  
 218 latent space dynamic updates and with the geometric mappings.

219 The methods are organized in modules for use in different combinations to obtain variations of  
 220 the discussed formulations of Geometric Dynamic Variational Autoencoders (GD-VAEs). We  
 221 give some examples of how to use the package in some specific cases below and in the code  
 222 repository.

## 223 Example Package Usage

224 The GD-VAEs package can be used for learning representations of non-linear dynamics on  
 225 latent spaces having general geometry and topology. By accommodating general topologies,  
 226 the methods can help facilitate obtaining more parsimonious representations or in enhancing  
 227 training of encoders and decoders by restricting responses to smaller subsets of latent codes  
 228 and on lower dimensional spaces. To help benchmark the methods, they have been used  
 229 for learning representations for the non-linear dynamics of PDEs arising in physics, including  
 230 Burgers' equations and reaction-diffusion systems, and for constrained mechanical systems  
 231 in ([Lopez & Atzberger, 2022](#)). The methods for incorporating geometric and topological  
 232 information present opportunities to simplify model representations, aid in interpretability, and  
 233 enhance robustness of predictions in data-driven learning methods. The GD-VAEs can be used  
 234 to obtain representations for use with diverse types of learning tasks involving dynamics.

## 235 Example Codes for Setting Up GD-VAE Models and Training

236 The GD-VAEs package provides methods for specifying the three components (i) encoder type,  
 237 (ii) latent space type and geometry, and (iii) decoder type. This is done by instantiating objects  
 238 from the package for each of these model classes in PyTorch and then composing the outputs.  
 239 A typical way to set up a GD-VAE model and compute training gradients is given below.

```
240 # set up the model
241 phi = {}; # encoder
242 encoder = Encoder_Fully_Connected(encoder_size,latent_dim);
243 point_cloud_periodic_proj_with_time
244     = PointCloudPeriodicProjWithTime(num_points_in_cloud);
245 phi['model_mu']
246     = lambda input : point_cloud_periodic_proj_with_time(encoder.mean(input));
247 phi['model_log_sigma_sq'] = encoder.log_variance;
248
249 latent_map = model_utils.latent_map_forward_in_time; # evolution map forward in time
250 latent_map_params = {'time_step':time_step};
251
252 theta = {}; # decoder
253 decoder = Decoder_Fully_Connected(decoder_size,latent_dim);
254 theta['model_mu'] = decoder.mean;
```



```

309         'params_klein':params_klein,
310         'device':device}});
311
312     # create the mapping to the manifold
313     manifold_map = gd_vae.geo_map.ManifoldPointCloudLayer(params_map);
314
315     return manifold_map;

```

316 For other latent space maps and additional details, see the codes in the package. We now give an  
 317 overview of a few example applications of GD-VAEs for learning dynamics and representations.

### 318 Non-linear Burgers' PDE: Learning Representations and Predicting Dynamics

319 We consider reductions for the non-linear Burgers' Equation

$$u_t = -uu_x + \nu u_{xx}, \quad u(0, x; \alpha) = q(x; \alpha).$$

320 The  $q(x; \alpha)$  gives a parameterized family of initial conditions. We consider the three cases  
 321 when the initial conditions are parameterized as

$$u(x, t; \alpha) = \alpha \sin(2\pi x) + (1 - \alpha) \cos^3(2\pi x), \quad \alpha \in [0, 1],$$

322 parameterized periodically as

$$u_\alpha(x, t=0) = \begin{bmatrix} \cos(2\pi\alpha) \\ \sin(2\pi\alpha) \end{bmatrix} \cdot \begin{bmatrix} \cos(2\pi x) \\ \sin(2\pi x) \end{bmatrix}, \quad \alpha \in [0, 1],$$

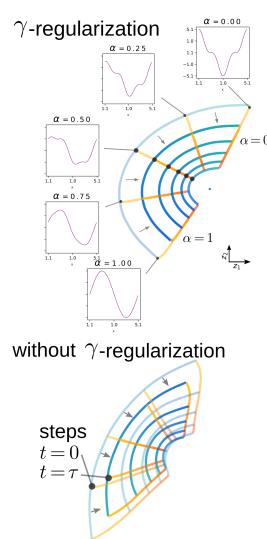
323 and parameterized as doubly-periodic as

$$u_{\alpha_1, \alpha_2}(x, 0) = \begin{bmatrix} \cos(\alpha_1) \\ \sin(\alpha_1) \\ \cos(\alpha_2) \\ \sin(\alpha_2) \end{bmatrix} \cdot \begin{bmatrix} \cos(2\pi x) \\ \sin(2\pi x) \\ \cos(4\pi x) \\ \sin(4\pi x) \end{bmatrix}, \quad \alpha_1, \alpha_2 \in [0, 2\pi].$$

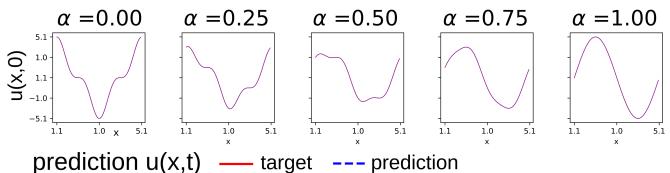
324 When learning the representations for the non-linear dynamics, an important role is played  
 325 by the reconstruction regularization  $\mathcal{L}_{RR}$ . This helps to ensure that the latent space not  
 326 only can make predictions of future observations but also reconstruct the current state of  
 327 the system. This is important to help ensure the learned representations can be used for  
 328 multi-step predictions through composition of subsequent predictions to generate a sequence of  
 329 states. Without the reconstruction regularization the learned representations may be misaligned  
 330 between time-steps where only single steps predictions are reliable. We show the impact in  
 331 multi-step predictions in the Figures. We also show results for this regularization when varying  
 332  $\gamma$  in the Tables.

Burgers' Equation:  $u_t = -uu_x + \nu u_{xx}$

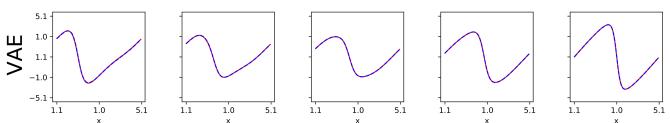
evolution in latent space



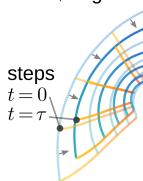
initial condition  $u(x,0)$



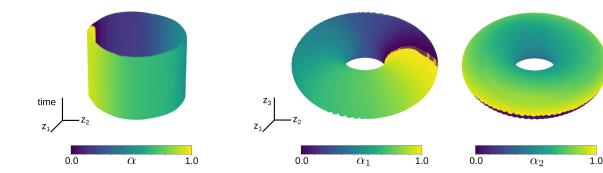
prediction  $u(x,t)$  — target — prediction



without  $\gamma$ -regularization



manifold latent spaces (periodic cases)



**Figure 6:** Burgers' Equation: Learning Representations for the Non-Linear Dynamics. The encodings learned by GD-VAEs is shown for the collection of solutions associated with the initial conditions  $u(x,0;\alpha)$ . This includes the cases with and without the reconstruction regularization  $\mathcal{L}_{RR}$ , which is seen to be important for consistency for multi-step predictions. For periodic and doubly-periodic initial conditions, manifold latent spaces are used. The predictions of the learned dynamics are shown for different values of  $\alpha$ .

When the underlying parameterized PDE has periodic parameters, this prior information can be incorporated by learning representations over manifold latent spaces. A single periodic structure would map the dynamics to the surface of a cylinder and doubly periodic can be mapped to the surface of a product space of a torus cross  $\mathbb{R}^1$ . In practice, we use a Clifford Torus (flat torus) in  $\mathbb{R}^4$ . We show results for the learned representations for single and doubly periodic cases and how they compare to other VAE and AE methods in the Tables.

Periodic: Cylinder Topology.				
Method	Dim	0.00s	0.50s	1.00s
GD-VAE	3	$2.12e-02 \pm 9.3e-05$	$2.57e-02 \pm 2.7e-03$	$4.72e-02 \pm 5.5e-03$
VAE-3D	3	$2.32e-02 \pm 3.8e-03$	$2.98e-02 \pm 3.4e-03$	$5.67e-02 \pm 5.6e-03$
AE (no projection)	3	$1.39e-02 \pm 4.6e-04$	$8.55e-02 \pm 5.9e-03$	$3.18e-01 \pm 1.3e-02$
Doubly-Periodic: Torus Topology.				
Method	Dim	0.00s	0.50s	1.00s
GD-VAE	5	$5.48e-02 \pm 3.8e-03$	$7.32e-02 \pm 5.8e-03$	$1.55e-01 \pm 1.0e-02$
VAE-5D	5	$5.56e-02 \pm 1.3e-03$	$6.73e-02 \pm 2.7e-03$	$1.42e-01 \pm 5.7e-03$
AE (no projection)	5	$2.80e-02 \pm 5.4e-04$	$1.84e-01 \pm 6.9e-03$	$8.04e-01 \pm 6.3e-02$

**Figure 7:** Accuracy of Predictions

The GD-VAEs are able to learn representations for making accurate multi-step predictions. The ability to incorporate known topological information into the latent space representations is found to help enhance the stability and accuracy of the learned dynamics.

### 342 Constrained Mechanical Systems: Learning Representations

343 We show how the GD-VAEs can be used for learning mappings to manifolds to represent  
344 constrained degrees of freedom. We consider the example of a basic arm actuator described by  
345 two locations in  $\mathbb{R}^2$ ,  $x_1, x_2$ . When the arm length is constrained to be rigid, this maps to a  
346 doubly periodic system which can be represented by states on the surface of a torus. We also  
347 consider a more exotic constraint where the actuator lengths are constrained so the points lie  
348 on a Klein bottle in  $\mathbb{R}^4$ . We show results for the representations that GD-VAEs learns in the  
349 Figures.

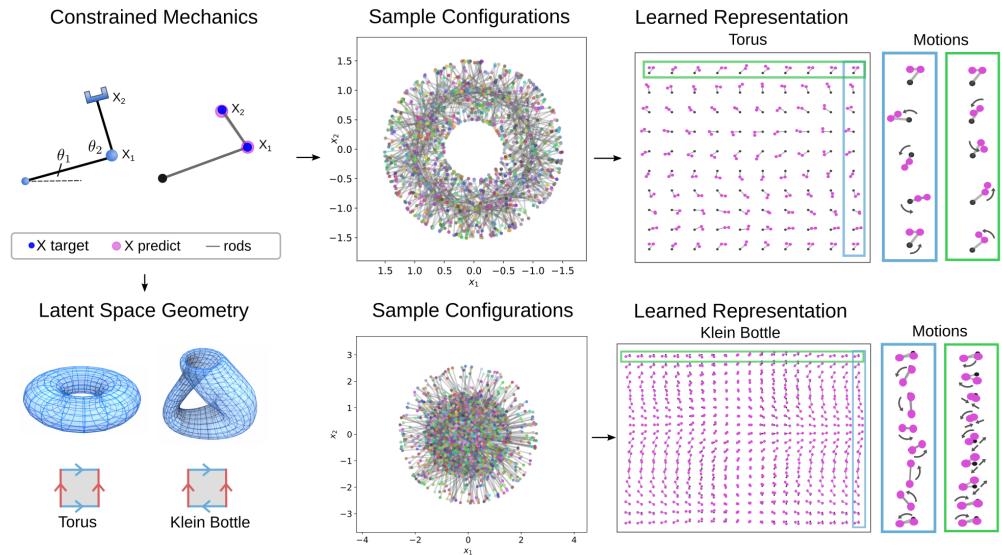


Figure 8: Example Code Usage

350 These results show that GD-VAEs is able to learn representations of systems with geometric  
351 constraints. This example also shows how non-orientable manifolds can be used for latent  
352 spaces in learned representations. For additional discussions see ([Lopez & Atzberger, 2022](#)).

### 353 Spatial-Temporal Reaction-Diffusion Systems: Learning Representations and 354 Predicting Dynamics

355 We consider the non-linear dynamics of spatial-temporal fields governed by the reaction-diffusion  
356 system

$$\frac{\partial u}{\partial t} = D_1 \Delta u + f(u, v), \quad \frac{\partial v}{\partial t} = D_2 \Delta v + g(u, v).$$

357 The  $u = u(x, t)$  and  $v = v(x, t)$  give the spatially distributed concentration of each chemical species  
358 at time  $t$  with  $x \in \mathbb{R}^2$ . We consider the case with periodic boundary conditions with  
359  $x \in [0, L] \times [0, L]$ . We develop learning methods for investigating the Brusselator system  
360 ([Prigogine & Lefever, 1968](#); [Prigogine & Nicolis, 1967](#)), which is known to have regimes  
361 exhibiting limit cycles ([Hirsch et al., 2012](#); [Strogatz, 2018](#)). This indicates after an initial  
362 transient the orbit associated with the dynamics will approach localizing near a subset of states  
363 having a geometric structure topologically similar to a circle. We show how GD-VAE can  
364 utilize this topological information to construct latent spaces for encoding states of the system.  
365 The Brusselator ([Prigogine & Lefever, 1968](#); [Prigogine & Nicolis, 1967](#)) has reactions with  
366  $f(u, v) = a - (1 + b)u + vu^2$  and  $g(u, v) = bu - vu^2$ . We take throughout the diffusivity  
367  $D_1 = 1, D_2 = 0.1$  and reaction rates  $a = 1, b = 3$ .

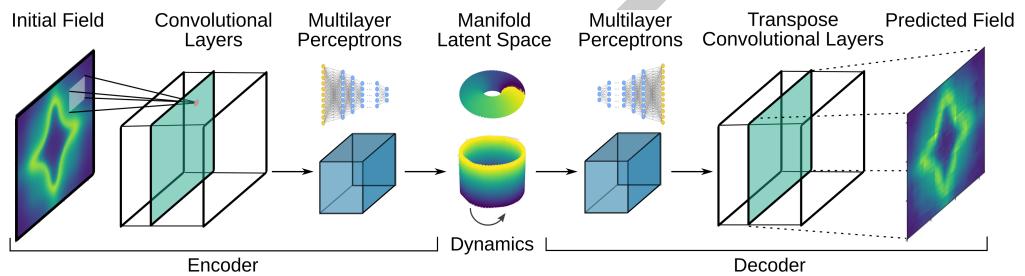
368 We specify a manifold latent space having the geometry of a cylinder  $\mathcal{M} = \mathcal{B} \times \mathbb{R}$  with  
369  $\mathcal{B} = S^1$  and axis in the  $z_3$ -direction. We prescribe on this latent space the dynamics having

370 the rotational evolution

$$\mathbf{z}(t + \Delta t) = \begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) & 0 \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{z}(t).$$

371 This is expressed in terms of an embedding in  $\mathbb{R}^3$ . The  $\omega$  gives the angular velocity. This  
 372 serves to regularize how the encoding of the reaction-diffusion system is organized in latent  
 373 space.

374 The spatial-temporal fields are handled using encoders and decoders based on Convolutional  
 375 Neural Networks (CNNs) and Transpose-CNNs (T-CNNs), see the Figures. For additional  
 376 discussions see ([Lopez & Atzberger, 2022](#)).



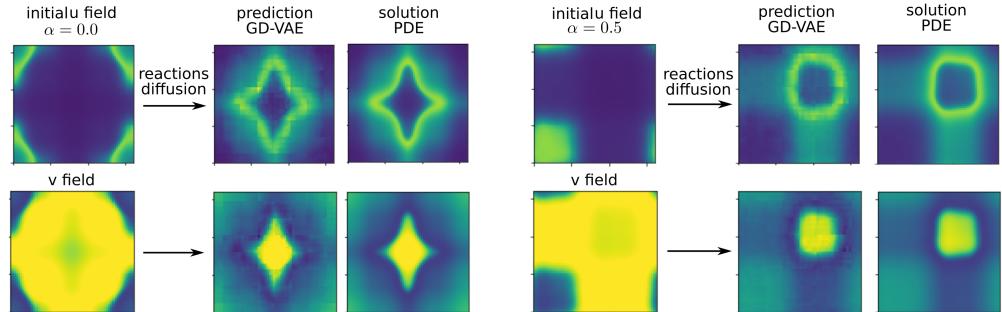
**Figure 9:** Non-Linear Dynamics of Brusselator: Learning Representations

377 We show the accuracy of the GD-VAEs in predicting multiple steps of the dynamics in the  
 378 Tables. We compare the GD-VAEs' predictions with those of more standard VAEs and AEs.  
 379 We find incorporating the known topological information improves the multi-step prediction  
 380 accuracy.

Method	Dim	0.00s	4.00s	8.00s
GD-VAE	3	$3.16e-02 \pm 5.4e-03$	$3.63e-02 \pm 7.0e-03$	$3.56e-02 \pm 8.3e-03$
VAE-3D	3	$2.61e-02 \pm 2.9e-03$	$2.08e-01 \pm 1.8e-01$	$2.16e-01 \pm 1.9e-01$
AE (no projection)	3	$2.36e-02 \pm 1.8e-03$	$3.49e-01 \pm 3.2e-01$	$1.55e-01 \pm 1.3e-01$

**Figure 10:** Accuracy of Predictions

381 We show some predictions of the spatial fields of the GD-VAEs in comparison with solving the  
 382 reaction-diffusion system using a PDE solver in the Figures.



**Figure 11:** Non-Linear Dynamics of Brusselator: Predicting Dynamics

383 The GD-VAEs with CNN encoder and T-CNN decoders were found to be able to make multi-  
 384 step predicts for the reaction-diffusion dynamics and concentration fields  $u, v$  comparable to

385 the PDE solver solutions. The ability of the GD-VAEs to incorporate into the latent space  
386 representations known topological information allows for improving the stability and accuracy  
387 of the learned representations of the non-linear dynamics.

## 388 Conclusion

389 The package GD-VAEs provides machine learning methods for data-driven learning of models  
390 for non-linear dynamics using both standard and manifold latent spaces. We describe here  
391 our initial implementation. For updates, examples, and additional information please see  
392 <https://github.com/gd-vae/gd-vae/> and <http://atzberger.org/>.

## 393 Acknowledgements

394 Authors research supported by grants DOE Grant ASCR PHILMS DE-SC0019246, NSF Grant  
395 DMS-1616353. Authors also acknowledge UCSB Center for Scientific Computing NSF MRSEC  
396 (DMR1121053) and UCSB MRL NSF CNS-1725797. P.J.A. would also like to acknowledge a  
397 hardware grant from Nvidia.

## 398 References

- 399 Archer, E., Park, I. M., Buesing, L., Cunningham, J., & Paninski, L. (2015). Black box  
400 variational inference for state space models. *arXiv Preprint arXiv:1511.07367*. <https://doi.org/10.48550/arXiv.1511.07367>
- 402 Atzberger, P. J. (2023). GD-VAE package v1.0.0. *Zenodo*. <https://doi.org/10.5281/zenodo.7945271>
- 404 Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state  
405 markov chains. *Ann. Math. Statist.*, 37(6), 1554–1563. <https://doi.org/10.1214/aoms/1177699147>
- 407 Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review  
408 for statisticians. *Journal of the American Statistical Association*, 112(518), 859–877.  
409 <https://doi.org/10.1080/01621459.2017.1285773>
- 410 Carlsson, G., Ishkhanov, T., Silva, V. de, & Zomorodian, A. (2008). On the local behavior  
411 of spaces of natural images. *International Journal of Computer Vision*, 76(1), 1–12.  
412 <https://doi.org/10.1007/s11263-007-0056-x>
- 413 Chatterjee, A. (2000). An introduction to the proper orthogonal decomposition. *Current  
414 Science*, 78(7), 808–817. <http://www.jstor.org/stable/24103957>
- 415 Chiuso, A., & Pillonetto, G. (2019). System identification: A machine learning perspective.  
416 *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1), 281–304. <https://doi.org/10.1146/annurev-control-053018-023744>
- 418 Cover, T. M., & Thomas, J. A. (2006). *Elements of information theory (wiley series in  
419 telecommunications and signal processing)*. Wiley-Interscience. <https://doi.org/10.1002/047174882X>
- 421 Das, S., & Giannakis, D. (2019). *Delay-coordinate maps and the spectra of koopman operators*.  
422 175, 1107–1145. <https://doi.org/10.1007/s10955-019-02272-w>
- 423 Del Moral, P. (1997). Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de  
424 l'Académie Des Sciences - Series I - Mathematics*, 325(6), 653–658. [https://doi.org/10.1016/S0764-4442\(97\)84778-7](https://doi.org/10.1016/S0764-4442(97)84778-7)

- 426 Ghahramani, Z., & Roweis, S. T. (1998). Learning nonlinear dynamical systems using an  
 427 EM algorithm. In M. J. Kearns, S. A. Solla, & D. A. Cohn (Eds.), *Advances in neural*  
 428 *information processing systems 11, [NIPS conference, denver, colorado, USA, november*  
 429 *30 - december 5, 1998]* (pp. 431–437). The MIT Press. <https://proceedings.neurips.cc/paper/1998/hash/0ebcc77dc72360d0eb8e9504c78d38bd-Abstract.html>
- 431 Godsill, S. (2019). Particle filtering: The first 25 years and beyond. *Proc. Speech and*  
 432 *Signal Processing (ICASSP) ICASSP 2019 - 2019 IEEE Int. Conf. Acoustics*, 7760–7764.  
 433 <https://doi.org/10.1109/ICASSP.2019.8683411>
- 434 Gross, B. J., Trask, N., Kuberry, P., & Atzberger, P. J. (2020). Meshfree methods on manifolds  
 435 for hydrodynamic flows on curved surfaces: A generalized moving least-squares (GMLS)  
 436 approach. *Journal of Computational Physics*, 409, 109340. <https://doi.org/10.1016/j.jcp.2020.109340>
- 438 Hirsch, M. W., Smale, S., & Devaney, R. L. (2012). *Differential equations, dynamical systems,*  
 439 *and an introduction to chaos*. Academic press. <https://doi.org/10.1016/C2009-0-61160-0>
- 440 Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal*  
 441 *of Basic Engineering*, 82(1), 35–45. <https://doi.org/10.1115/1.3662552>
- 442 Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. *2nd International*  
 443 *Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014,*  
 444 *Conference Track Proceedings*. <http://arxiv.org/abs/1312.6114>
- 445 Korda, M., Putinar, M., & Mezic, I. (2020). Data-driven spectral analysis of the koopman  
 446 operator. *Applied and Computational Harmonic Analysis*, 48(2), 599–629. <https://doi.org/10.1016/j.acha.2018.08.002>
- 448 Krishnan, R. G., Shalit, U., & Sontag, D. A. (2017). Structured inference networks for nonlinear  
 449 state space models. In S. P. Singh & S. Markovitch (Eds.), *Proceedings of the thirty-first*  
 450 *AAAI conference on artificial intelligence, february 4-9, 2017, san francisco, california, USA*  
 451 (pp. 2101–2109). AAAI Press. <https://doi.org/10.1609/aaai.v31i1.10779>
- 452 Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Statist.*,  
 453 22(1), 79–86. <https://doi.org/10.1214/aoms/1177729694>
- 454 Kutz, J. N., Brunton, S. L., Brunton, B. W., & Proctor, J. L. (2016). *Dynamic mode*  
 455 *decomposition*. Society for Industrial; Applied Mathematics. <https://doi.org/10.1137/1.9781611974508>
- 457 Lopez, R., & Atzberger, P. J. (2022). GD-VAEs: Geometric dynamic variational autoencoders  
 458 for learning nonlinear dynamics and dimension reductions. *arXiv Preprint arXiv:2206.05183*.  
 459 <https://arxiv.org/abs/2206.05183>
- 460 Mendez, M. A., Balabane, M., & Buchlin, J. M. (2018). *Multi-scale proper orthogonal*  
 461 *decomposition (mPOD)*. <https://doi.org/10.1063/1.5043720>
- 462 Mezić, I. (2013). Analysis of fluid flows via spectral properties of the koopman operator. *Annual Review of Fluid Mechanics*, 45(1), 357–378. <https://doi.org/10.1146/annurev-fluid-011212-140652>
- 465 Nelles, O. (2013). *Nonlinear system identification: From classical approaches to neural*  
 466 *networks and fuzzy models*. Springer Science & Business Media. <https://doi.org/10.1007/978-3-662-04323-3>
- 468 Pawar, S., Ahmed, S. E., San, O., & Rasheed, A. (2020). *Data-driven recovery of hidden physics*  
 469 *in reduced order modeling of fluid flows*. 32, 036602. <https://doi.org/10.1063/5.0002051>
- 470 Prigogine, I., & Lefever, R. (1968). Symmetry breaking instabilities in dissipative systems. II.  
 471 *The Journal of Chemical Physics*, 48(4), 1695–1700. <https://doi.org/10.1063/1.1668896>

- 472 Prigogine, I., & Nicolis, G. (1967). On symmetry-breaking instabilities in dissipative systems.  
473 *The Journal of Chemical Physics*, 46(9), 3542–3550. <https://doi.org/10.1063/1.1841255>
- 474 Raissi, M., & Karniadakis, G. E. (2018). Hidden physics models: Machine learning of  
475 nonlinear partial differential equations. *Journal of Computational Physics*, 357, 125–141.  
476 <https://arxiv.org/abs/1708.00588>
- 477 Saul, L. K. (2020). A tractable latent variable model for nonlinear dimensionality reduction.  
478 *Proceedings of the National Academy of Sciences*, 117(27), 15403–15408. <https://doi.org/10.1073/pnas.1916012117>
- 480 Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data.  
481 *Journal of Fluid Mechanics*, 656, 5–28. <https://doi.org/10.1017/S0022112010001217>
- 482 Schmidt, M., & Lipson, H. (2009). *Distilling free-form natural laws from experimental data*.  
483 324, 81–85. <https://doi.org/10.1126/science.1165893>
- 484 Schoukens, J., & Ljung, L. (2019). Nonlinear system identification: A user-oriented road  
485 map. *IEEE Control Systems Magazine*, 39(6), 28–99. <https://doi.org/10.1109/MCS.2019.2938121>
- 487 Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Gorenne, P.-Y., Hjalmarsson,  
488 H., & Juditsky, A. (1995). Nonlinear black-box modeling in system identification: A  
489 unified overview. *Automatica*, 31(12), 1691–1724. [https://doi.org/10.1016/0005-1098\(95\)00120-8](https://doi.org/10.1016/0005-1098(95)00120-8)
- 491 Strogatz, S. H. (2018). *Nonlinear dynamics and chaos: With applications to physics, biology,  
492 chemistry, and engineering*. CRC press. <https://doi.org/10.1201/9780429492563>
- 493 Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L., & Kutz, J. N. (2014).  
494 On dynamic mode decomposition: Theory and applications. *Journal of Computational  
495 Dynamics*. <https://doi.org/10.3934/jcd.2014.1.391>
- 496 Van Der Merwe, R., Doucet, A., De Freitas, N., & Wan, E. (2000). The unscented par-  
497 ticle filter. *Proceedings of the 13th International Conference on Neural Information  
498 Processing Systems*, 563–569. [https://proceedings.neurips.cc/paper\\_files/paper/2000/file/f5c3dd7514bf620a1b85450d2ae374b1-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2000/file/f5c3dd7514bf620a1b85450d2ae374b1-Paper.pdf)
- 500 Wan, E. A., & Van Der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation.  
501 *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications,  
502 and Control Symposium (Cat. No.00EX373)*, 153–158. <https://doi.org/10.1109/ASSPCC.2000.882463>
- 504 Whitney, H. (1944). The self-intersections of a smooth n-manifold in 2n-space. *Annals of  
505 Mathematics*, 45(2), 220–246. <https://doi.org/10.2307/1969265>