# Appendix

## A DyLand Details

In this section, we illustrate the derivations of the variational bound in DyLand sequentially, and present other details about model construction and training.

### Dynamic Manifold Learning $p(\mathbf{U}|\mathbf{V},\mathbf{S})$

Algorithm 1 summarizes the training process of our proposed dynamic manifold learning, which has been described in main text. After model converges, $\mathbf{U}$ is obtained via line 13.

The time complexity of computing the unbiased stochastic estimator of the likelihood is $\mathcal{O}(d)$, where $d$ is the dimension of $\mathbf{V}$ (Hutchinson 1989). Evaluating $f$ is $\mathcal{O}(dC)$, where $C$ is the largest size of hidden layers. Overall, the complexity of computing likelihood in Algorithm 1 is $\mathcal{O}((dC+d)L)$, where $L$ is the number of evaluations of $f$ used by the ODE solver (Grathwohl et al. 2018).

### Approximating $p(\mathbf{Y},\mathbf{W}|\mathbf{A},\mathbf{V},\mathbf{S})$

Let $\boldsymbol{\Omega}=\{\mathbf{A},\mathbf{V},\mathbf{S}\}$ for simplicity. Existing methods model $p(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega})$ directly, however, ignoring the prior distributions of $\mathbf{Y}$ and $\mathbf{W}$ (discussed in section C). Thus, we propose a variational framework to approximate the intractable $p(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega})$ by $q_\rho(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega})$, and the KL-divergence of the joint distribution is derived as follows:

$$
\begin{aligned}
&D_{\mathrm{KL}}(q_\rho(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega})\|p(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega}))\\
=&\mathbb{E}_{\mathbf{Y},\mathbf{W}\sim q_\rho}[\log\frac{q_\rho(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega})}{p(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega})}]\\
=&\mathbb{E}_{\mathbf{Y},\mathbf{W}\sim q_\rho}[\log\frac{q_\varphi(\mathbf{W}|\boldsymbol{\Omega})q_\phi(\mathbf{Y}|\mathbf{W},\boldsymbol{\Omega})}{p(\mathbf{W}|\boldsymbol{\Omega})p(\mathbf{Y}|\mathbf{W},\boldsymbol{\Omega})}]\\
=&\mathbb{E}_{\mathbf{W}\sim q_\varphi}[\log\frac{q_\varphi(\mathbf{W}|\boldsymbol{\Omega})}{p(\mathbf{W}|\boldsymbol{\Omega})}]+\mathbb{E}_{\mathbf{Y}\sim q_\phi}[\log\frac{q_\phi(\mathbf{Y}|\mathbf{W},\boldsymbol{\Omega})}{p(\mathbf{Y}|\mathbf{W},\boldsymbol{\Omega})}]\\
=&D_{\mathrm{KL}}(q_\varphi(\mathbf{W}|\boldsymbol{\Omega})\|p(\mathbf{W}|\boldsymbol{\Omega}))\\
&+D_{\mathrm{KL}}(q_\phi(\mathbf{Y}|\mathbf{W},\boldsymbol{\Omega})\|p(\mathbf{Y}|\mathbf{W},\boldsymbol{\Omega})),
\end{aligned}
\tag{A.1}
$$

where $\rho$, $\varphi$ and $\phi$ are parameters of $q(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega})$, $q(\mathbf{W}|\boldsymbol{\Omega})$ and $q(\mathbf{Y}|\mathbf{W},\boldsymbol{\Omega})$ respectively, and denoted as $q_\rho$, $q_\varphi$ and $q_\phi$ for simplicity. From Eq.(A.1), we find that $p(\mathbf{Y},\mathbf{W}|\boldsymbol{\Omega})$ can be approximated by the KL-divergences of the marginal distributions respectively, and we introduce them in detail subsequently.



Algorithm 1: Dynamic manifold learning $p(\mathbf{U}|\mathbf{V},\mathbf{S})$

**Input:** Locations $\mathbf{V}\in\mathbb{R}^{N\times d}$, deformations $\mathbf{S}\in\mathbb{R}^{N\times T}$, manifold mapping $f$, start time $t_0$, stop time $t_1$, Hutchinson's estimator $\mathbb{E}_H$ (Hutchinson 1989), predetermined ODE solver.

1: **function** $f_{aug}^{-1}([\mathbf{u}^t,\mathbf{s}],t)$:
▷ Define augment $f$ with trace estimation
2: $\mathrm{Tr}=-\mathbb{E}_H(f^{-1},\mathbf{u}^t)$; ▷ Estimate trace
3: $f_t=f^{-1}(\mathbf{u}^t,\mathbf{s},t)$; ▷ Transform the input by $f^{-1}$
4: **return** $[f_t,\mathrm{Tr}]$
5: **end function**
6: Initialize $\tau=1,i=1,\mathcal{L}_{\mathbf{U}}=0$;
7: **while** $\tau\le T$ **do**
8: **while** $i\le N$ **do**
9: $\mathbf{u}^{t_1}=\mathbf{v}_i+\mathbf{s}_i^\tau$; ▷ Perturbe locations
10: $[\mathbf{u}^{t_0},\mathrm{Tr}]=\mathrm{Solver}(f_{aug}^{-1},[\mathbf{u}^{t_1},\mathbf{s}_i^\tau],t_0,t_1)$;
▷ Solve the ODE
11: $\log\hat{p}(\mathbf{u}^{t_1}|\mathbf{s}_i^\tau)=\log p(\mathbf{u}^{t_0})-\mathrm{Tr}$ ;
▷ Estimate log density via Eq.(7)
12: $\mathcal{L}_{\mathbf{U}}=\mathcal{L}_{\mathbf{U}}-\log\hat{p}(\mathbf{u}^{t_1}|\mathbf{s}_i^\tau)$; ▷ Add up loss
13: $\mathbf{u}_i^\tau=\mathbf{u}^{t_0}$; ▷ Get the manifold embedding
14: **end while**
15: **end while**
16: Minimize $\mathcal{L}_{\mathbf{U}}$ via Adam optimizer (Kingma and Ba 2017).

### Approximating $p(\mathbf{W}|\mathbf{A},\mathbf{V},\mathbf{S})$

In DyLand, $p(\mathbf{W}|\boldsymbol{\Omega})$ is approximated by $q_\varphi(\mathbf{W}|\boldsymbol{\Omega})$, and the parameters can optimized by the evidence lower bound (ELBO):

$$
\begin{aligned}
&\log p(\mathbf{A},\mathbf{S})-D_{\mathrm{KL}}(q_\varphi(\mathbf{W}|\mathbf{A},\mathbf{S})\|p(\mathbf{W}|\mathbf{A},\mathbf{S}))\\
=&\mathbb{E}_{q_\varphi}[\log p(\mathbf{A},\mathbf{S}|\mathbf{W})]-D_{\mathrm{KL}}(q_\varphi(\mathbf{W}|\mathbf{A},\mathbf{S})\|p(\mathbf{W})).
\end{aligned}
\tag{A.2}
$$

where $\mathbf{V}\perp\!\!\!\perp\mathbf{W}|\mathbf{A}$ and $p(\mathbf{W}|\boldsymbol{\Omega})=p(\mathbf{W}|\mathbf{A},\mathbf{S})$. Instead of approximating standard Gaussian $\mathcal{N}(\mathbf{0},\boldsymbol{I}^2)$ conventionally, we argue that, $\mathbf{W}$ should maintain a similar distribution to

$\mathbf{S}$, where it is aggregated from:

$$\begin{aligned}
&D_{\mathrm{KL}}\left(q_{\varphi}\left(\mathbf{W}^{\tau}|\mathbf{A}^{\tau},\mathbf{S}^{\tau}\right)\|p\left(\mathbf{W}^{\tau}\right)\right)\\
\simeq&D_{\mathrm{KL}}\left(q_{\varphi}\left(\mathbf{W}^{\tau}|\mathbf{A}^{\tau},\mathbf{S}^{\tau}\right)\|\mathcal{N}\left(\boldsymbol{\mu}^{\tau},{\boldsymbol{\sigma}^{\tau}}^{2}\right)\right)\\
=&D_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{W}},\boldsymbol{\sigma}_{\mathbf{W}}^{2}\right)\|\mathcal{N}\left(\boldsymbol{\mu}^{\tau},{\boldsymbol{\sigma}^{\tau}}^{2}\right)\right)\\
=&\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(2\log\frac{\sigma_i^j}{{\sigma_{\mathbf{w}}}_i^j}+\frac{\left(\mu_i^j-{\mu_{\mathbf{w}}}_i^j\right)^2}{{\sigma_i^j}^2}+\frac{{\sigma_{\mathbf{w}}}_i^{j\,2}}{{\sigma_i^j}^2}-1\right),
\end{aligned}\tag{A.3}$$

where $M$ is the dimension of the latent variable, and $\mathbf{W}^{\tau}\in\mathbb{R}^{N\times M}$. In practice, $\mathbf{W}^{\tau}$ can be extracted from dynamic adjacency $\mathbf{A}^{\tau}$ and $\mathbf{S}^{\tau}$ by a reparameterization trick as follows:

$$\mathbf{W}^{\tau}=\mathrm{GNN}_{\boldsymbol{\mu}_{\mathbf{W}}}\left(\mathbf{A}^{\tau},\mathbf{S}^{\tau}\right)+\exp\left(\mathrm{GNN}_{\boldsymbol{\sigma}_{\mathbf{W}}}\left(\mathbf{A}^{\tau},\mathbf{S}^{\tau}\right)\right)*\boldsymbol{\varepsilon},\tag{A.4}$$

where $\mathbf{W}^{\tau}\sim\mathcal{N}(\boldsymbol{\mu}_{\mathbf{W}},\boldsymbol{\sigma}_{\mathbf{W}}^{2})$ and $\boldsymbol{\varepsilon}$ is sampled from standard Gaussian. We used $\exp$ to learn log standard deviation as a real number. Note that, Eq.(A.4) is implemented by a variational graph auto-encoder (VGAE) (Kipf and Welling 2016).

The reconstruction term (a.k.a the decoder) in Eq.(A.2) can be estimated by Monte Carlo as:

$$\mathbb{E}_{q_{\varphi}}[\log p(\mathbf{A}^{\tau},\mathbf{s}^{\tau}|\mathbf{W}^{\tau})]\simeq\frac{1}{N}\sum_{i=1}^{N}\log p_{\xi}(\hat{s}_i^{\tau}|\mathbf{w}_i^{\tau}).\tag{A.5}$$

For simplicity, $p_{\xi}$ in Eq.(A.5) is a fully connected neural network and we choose graph convolutional network (GCN) (Kipf and Welling 2017) in Eq.(A.4). Note that other GNNs may improve the performance, which, however, is out of our focus.

Algorithm 2 summarizes the structure and training of the used VGAE, in which, Eqs.(A.3) and (A.5) of different time stamps are summarized together in addition to Eq.(15).

Algorithm 2: Training process of estimating $p(\mathbf{W}|\mathbf{A},\mathbf{V},\mathbf{S})$

**Input:** Adjacency matrix $\mathbf{A}\in\mathbb{R}^{N\times N}$, deformations $\mathbf{S}\in\mathbb{R}^{N\times T}$, latent shape $M$.
**Output:** Spatio-temporal features $\mathbf{W}\in\mathbb{R}^{N\times M}$.

1: $\boldsymbol{\mu}_{\mathbf{W}}=\mathrm{GCN}_{\boldsymbol{\mu}_{\mathbf{W}}}(\mathbf{A},\mathbf{S})$; ▷ Get mean
2: $\log\boldsymbol{\sigma}_{\mathbf{W}}=\mathrm{GCN}_{\boldsymbol{\sigma}_{\mathbf{W}}}(\mathbf{A},\mathbf{S})$; ▷ Get log standard deviation
3: Sample $\boldsymbol{\varepsilon}$ from $\mathcal{N}(\mathbf{0},\boldsymbol{I}^2)$;
4: $\mathbf{W}=\boldsymbol{\mu}_{\mathbf{W}}+\boldsymbol{\sigma}_{\mathbf{W}}*\boldsymbol{\varepsilon}$; ▷ Get $\mathbf{W}$ via reparameterization
5: $\widehat{\mathbf{S}}=\mathrm{FC}(\mathbf{W})$; ▷ Reconstruct $\widehat{\mathbf{S}}$ via Eq.(A.5)
6: Compute encoder loss by $\boldsymbol{\mu}_{\mathbf{W}}$ and $\boldsymbol{\sigma}_{\mathbf{W}}$ as Eq.(A.3), and add it to reconstruction loss to get ELBO in Eq.(A.2).

## Approximating $p\left(\mathbf{Y}|\mathbf{W},\mathbf{A},\mathbf{V},\mathbf{S}\right)$ via Neural ODEs

The second $D_{\mathrm{KL}}$ in Eq.(A.1) is derived as:

$$\begin{aligned}
&D_{\mathrm{KL}}\left(q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\boldsymbol{\Omega}^{\tau}\right)\|p\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\boldsymbol{\Omega}^{\tau}\right)\right)\\
=&D_{\mathrm{KL}}\left(q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)\|p\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)\right)\\
=&\log p(\mathbf{W}^{\tau},\mathbf{S}^{\tau})-\mathbb{E}_{q_{\phi}}[\log p(\mathbf{W}^{\tau},\mathbf{S}^{\tau}|\mathbf{Y}^{\tau})]\\
&+D_{\mathrm{KL}}\left(q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)\|p\left(\mathbf{Y}^{\tau}\right)\right)
\end{aligned}\tag{A.6}$$

Algorithm 3: Training of deformation dynamic $p\left(\mathbf{Y}|\mathbf{W},\boldsymbol{\Omega}\right)$

**Input:** Deformations $\mathbf{S}\in\mathbb{R}^{N\times T}$, spatio-temporal features $\mathbf{W}$, dynamic $g$, ground truth $\mathbf{Y}$, Hutchinson's estimator $\mathbb{E}_H$, predetermined ODE solver.

1: **function** $g_{aug}(\mathbf{z}^t,t)$:
▷ Define augment $g$ with trace estimation
2: Tr $=-\mathbb{E}_H(g,\mathbf{z}^t)$; ▷ Estimate trace
3: $g_t=g(\mathbf{z}^t,t)$; ▷ Transform the input by $g$
4: **return** $[g_t,$ Trace$]$
5: **end function**
6: Initialize $\tau=1,i=1,\iota=T+1$;
7: **while** $\iota\leq T'$ **do** ▷ Make predictions from $T+1$ to $T'$
8: $t_1=\iota$ ;
9: **while** $\tau\leq T$ **do** ▷ Predict on each observation
10: $t_0=\tau$;
11: **while** $i\leq N$ **do**
12: $\mathbf{z}^{t_0}=[\mathbf{s}_i^{\tau},\mathbf{w}_i^{\tau}]^{\intercal}$; ▷ Concatenate $\mathbf{s}_i^{\tau}$ and $\mathbf{w}_i^{\tau}$
13: $[\mathbf{z}^{t_1},\mathrm{Tr}]=\mathrm{Solver}(g_{aug},\mathbf{z}^{t_0},t_1,t_0)$;
▷ Solve the ODE
14: $\widetilde{\mathbf{Y}}_i^{\tau}=\mathbf{z}^{t_1}$; ▷ Get the prediction
15: **end while**
16: **end while**
17: **end while**
18: $\widehat{\mathbf{Y}}=\mathrm{FC}(\widetilde{\mathbf{Y}})$; ▷ Aggregate predictions $\widetilde{\mathbf{Y}}$
19: Calculate overall loss $\mathcal{L}_{\mathbf{Y}}=\mathcal{L}_{\mathrm{MSE}}(\widehat{\mathbf{Y}},\mathbf{Y})+\mathcal{L}_{\mathrm{ELBO}}(\widehat{\mathbf{Y}})$ in Eq.(15), and minimize $\mathcal{L}_{\mathbf{Y}}$ with Adam.

where the joint distribution is considered invariant. Note that the mapping from $\mathbf{W},\mathbf{S}$ to $\mathbf{Y}$ is a bijection, thus the reconstruction term is omitted. Overall, Eq.(A.6) is simplified as:

$$\begin{aligned}
&D_{\mathrm{KL}}\left(q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\boldsymbol{\Omega}\right)\|p\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\boldsymbol{\Omega}\right)\right)\\
\simeq&D_{\mathrm{KL}}\left(q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)\|p\left(\mathbf{Y}^{\tau}\right)\right)\\
=&\int q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)\log\frac{q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)}{p\left(\mathbf{Y}^{\tau}\right)}d\mathbf{Y}^{\tau}\\
=&\int q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)\log q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)d\mathbf{Y}^{\tau}\\
&-\int q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)\log p\left(\mathbf{Y}^{\tau}\right)d\mathbf{Y}^{\tau}\\
=&-H[q_{\phi}\left(\mathbf{Y}^{\tau}|\mathbf{W}^{\tau},\mathbf{S}^{\tau}\right)]-\mathbb{E}_{q_{\phi}}[\log p\left(\mathbf{Y}^{\tau}\right)]\\
\simeq&\sum_{i=1}^{N}\left(p\left(y_i^{\tau}\right)\log p\left(y_i^{\tau}\right)\right)-\frac{1}{N}\sum_{i=1}^{N}\log\mathcal{N}\left(y_i|\mu_i^{\tau},{\sigma_i^{\tau}}^2\right),
\end{aligned}\tag{A.7}$$

where $\log p\left(y\right)$ can be estimated by the Hutchinson's estimator. And we assume distribution of deformations at time $\tau$ and $\tau+1$ are the same.

Algorithm 3 summarizes the training of co-evolution of $\mathbf{S}$ and $\mathbf{W}$ as a dynamic system. After model converged, $\widehat{\mathbf{Y}}\in\mathbb{R}^{N\times T'}$ is obtained via line 18. The computation complexity is almost the same with $p(\mathbf{U}|\mathbf{V},\mathbf{S})$.

## Pre-training with Simulations

We propose to perturb $\mathbf{V}$ with simulated deformations, because training requires much more data than we have. More

specifically, $\mathbf{S}^\tau$ are the deformation observations on $N$ locations at the $\tau$ time stamp, and is considered as one complete input. But collecting observations is costly, and we only have about 70 observations in each dataset. Therefore, simulated perturbations are suggested to pre-train $\mathbf{v}_i' = f^{-1}(\mathbf{u}_i|\mathbf{s}_i^\tau)$.

We split the monitored area into $O$ grids, and randomly sample a perturbation vector $\boldsymbol{\epsilon}_i^o \sim \mathcal{N}(\boldsymbol{\mu}^o, \boldsymbol{\sigma}^{o2})$ for the $o$-th grid (Note that notations might be different from those in Section A), where $\boldsymbol{\mu}^o \sim unif[\boldsymbol{\mu} - \boldsymbol{\alpha}, \boldsymbol{\mu} + \boldsymbol{\alpha}]$ and $\boldsymbol{\sigma}^o \sim unif[\boldsymbol{\sigma} - \boldsymbol{\beta}_1, \boldsymbol{\sigma} + \boldsymbol{\beta}_2]$. Note that $\boldsymbol{\beta}_1 > \boldsymbol{\beta}_2$, because the variance of nearby neighbors is usually smaller. Then, we add the generated perturbation $\boldsymbol{\epsilon}_i^o$ to each monitored location $\mathbf{v}_i$ to simulate the deformation. Subsequently, we can estimate the deformation distribution of each location as $\mathbf{v}_i' = f^{-1}(\mathbf{u}_i|\boldsymbol{\epsilon}_i^o)$.

When $f$ has almost converged, we can train it with real data as described in the main body. After training is finished, we can use the real data to get an accurate mapping to the manifold space and obtain $\mathbf{U}^\tau$.

### Two-stage Training

The proposed DyLand has three main components: manifold learning, graph auto-encoder, and prediction. We propose to train our model in two stages rather a general end-to-end strategy, which can be argued in the following two aspects.

Begin with sufficient condition. $\mathbf{U}$ is a manifold and has realistic meanings, different from other latent representations in autoencoders of end-to-end tasks (Kingma and Welling 2014; Yin and Zhou 2018; Kipf and Welling 2016; Hasanzadeh et al. 2019). Therefore, manifold learning should have its own evaluation and optimization for interpretability. Necessary condition is that, variance bound Eq.(15) is not calculated by $\mathbf{A}$ directly, nor $\mathbf{U}$ and $\mathbf{V}$. Therefore, the optimization of DyLand is decided and only decided by Eqs.(A.3), (A.5) and (15).

Additionally, the two-stage strategy training is also time efficient. In practice, training of CNF is slow and it takes more than 30 hours to learn manifold on PBG-East with GTX 1070Ti. Finishing training of manifold first means it can monopolize memory and graphic processing units (GPUs), having bigger batchsize and quicker convergence and allowing us to try more prediction methods. Also, two-stage training strategy makes it easier to compare other manifold learning methods against ours.

## B Normalizing Flows

Since normalizing flow was proposed (Rezende and Mohamed 2015), it has gained much attention. The basic idea is to use a series of invertible transformations to learn a mapping from a simple distribution to a complicated and close-to-truth one. It has been successfully applied in many research areas including image generation (Kingma and Dhariwal 2018), natural language processing (Wang and Wang 2019), and manifold learning (Kim et al. 2020; Brehmer and Cranmer 2020; Rezende et al. 2020; Kuznetsov and Polykovskiy 2021).

Considering an invertible mapping $f : \mathbb{R}^d \Rightarrow \mathbb{R}^d$ with inverse $g = f^{-1}$, it transforms a random variable $\mathbf{z}$ with a simple distribution $p_{\mathbf{Z}}(\mathbf{z})$ (e.g., an isotropic Gaussian) that can generate a data point $\mathbf{x}$, i.e., $\mathbf{x} = f(\mathbf{z})$. The cumulative distribution of $\mathbf{x}$ and $\mathbf{z}$ can be modeled as:

$$F_{\mathbf{X}}(\mathbf{x}) = F_{\mathbf{Z}}(f^{-1}(\mathbf{x})) = \int_{-\infty}^{f^{-1}(\mathbf{x})} p_{\mathbf{Z}}(\mathbf{z})d\mathbf{z}. \tag{B.1}$$

By composing a series of transformations, the target $p_{\mathbf{X}}(\mathbf{x})$ can be arbitrarily complex, i.e., normalizing flow is an universal approximator (Rezende and Mohamed 2015; Kong and Chaudhuri 2020). Let $\mathbf{z}_i$ be the intermediate variables and transformations be $f_i$ $(i = 0, \ldots, k)$. The density $p_{\mathbf{X}}(\mathbf{x})$ obtained by $\mathbf{z}_0$ through $k$ times transformations is:

$$\mathbf{x} = f_k \circ \cdots \circ f_2 \circ f_1(\mathbf{z}_0), \tag{B.2}$$

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(\mathbf{z}) \prod_{k=1}^{K} \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|^{-1}. \tag{B.3}$$

Though looks nice in theory, computing the determinant of the Jacobian has a complexity of $O(kd^3)$. A few studies construct a tractable triangular Jacobian matrix to tackle this problem, which are called *triangular flows* (a.k.a, autoregressive flows) (Rezende and Mohamed 2015; Dinh, Sohl-Dickstein, and Bengio 2017; Huang et al. 2018; Jaini, Selby, and Yu 2019). Another line of research exploits the properties of matrix to construct well-designed invertible transformations with tractable non-triangular Jacobian called *non-triangular flows* (Tomczak and Welling 2017; Hasenclever et al. 2017; Behrmann et al. 2019; Chen et al. 2019). However, the expressive power of transforming arbitrary distributions is limited in some situations (Kong and Chaudhuri 2020), e.g., for high-dimensional data.

Recently, another type of infinite normalizing flows are proposed called continuous normalizing flow (CNF) (Chen et al. 2018b,a). The transformation in Eq.(B.2) is described as time-varying in CNF, and therefore can be solved by an ODE solver:

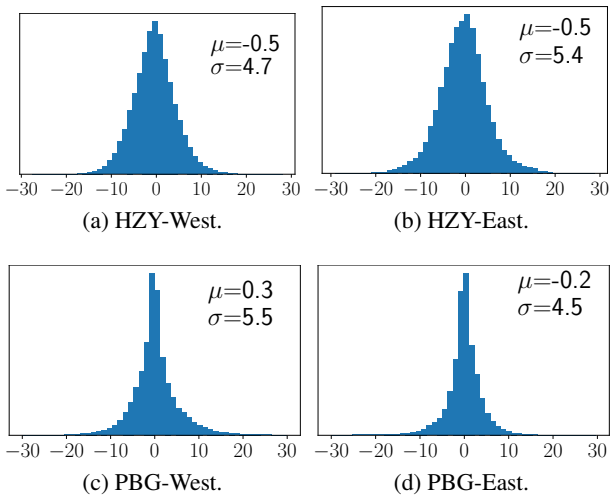
$$\begin{aligned} \frac{d\mathbf{h}_t}{dt} &= f(\mathbf{h}_t, t) \\ \mathbf{h}_{t_1} &= \mathbf{h}_{t_0} + \int_{t_0}^{t_1} f(\mathbf{h}_t, t)dt, \end{aligned} \tag{B.4}$$

where $\mathbf{h}_{t_0} = \mathbf{z}$ and $\mathbf{h}_{t_1} = \mathbf{x}$. The change in the log density in Eq.(B.3) can be computed as:

$$\begin{aligned} \frac{\partial \log p(\mathbf{h}_t)}{\partial t} &= -\operatorname{tr}\big(\frac{df}{d\mathbf{h}_t}\big) \\ \log p(\mathbf{h}_{t_1}) &= \log p(\mathbf{h}_{t_0}) - \int_{t_0}^{t_1} \operatorname{Tr}\big(\frac{df}{d\mathbf{h}_t}\big)dt. \end{aligned} \tag{B.5}$$

## C Experimental Details

In implementation of DyLand, We utilize the third party torchdiffeq [1] package (Chen et al. 2018b) to solve and back-propagate through ODEs.




(a) HZY-West. (b) HZY-East.

(c) PBG-West. (d) PBG-East.

Figure 1: Distribution of deformations.

### Dataset Property

The distribution of land deformations of the studied areas are shown in Figure 1. Clearly, the surface deformations follow Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$.

### Evaluation Protocols

We use following metrics to measure the land deformation prediction performance of models:

1. Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{N\Delta T}\sum_{\tau=T+1}^{T'}\sum_{i=1}^{N}(y_i^\tau - \hat{y}_i^\tau)^2}$$

2. Mean Absolute Error (MAE):

$$MAE = \frac{1}{N\Delta T}\sum_{\tau=T+1}^{T'}\sum_{i=1}^{N}|y_i^\tau - \hat{y}_i^\tau|$$

3. Accuracy:

$$Accuracy = \frac{1}{N}\sum_{\tau=T+1}^{T'}\sum_{i=1}^{N}\mathbf{I}\left(|y_i^\tau - \hat{y}_i^\tau| \leq \text{threshold}\right)$$

4. Coefficient of Determination ($R^2$):

$$R^2 = 1 - \frac{\sum_{\tau=T+1}^{T'}\sum_{i=1}^{N}(y_i^\tau - \hat{y}_i^\tau)^2}{\sum_{\tau=T+1}^{T'}\sum_{i=1}^{N}(y_i^\tau - \overline{\mathbf{Y}})^2}$$

5. Explained Variance Score (var):

$$var = 1 - \frac{var\{\mathbf{Y} - \widehat{\mathbf{Y}}\}}{Var\{\mathbf{Y}\}}$$

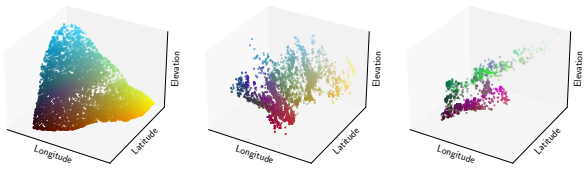
[1]https://github.com/rtqichen/torchdiffeq

where $\Delta T = T' - T$ is the range of predictions, $y_i^\tau$ and $\hat{y}_i^\tau$ represent the true and predicted measurement at time $\tau$ and $\mathbf{I}(x) = 1$ iff $x$ is true. Threshold for determining the correct prediction is 0.01. $\mathbf{Y}$, $\widehat{\mathbf{Y}}$ and $\overline{\mathbf{Y}}$ are real deformations, predicted deformations and average deformations, respectively.

## D More Experimental Results

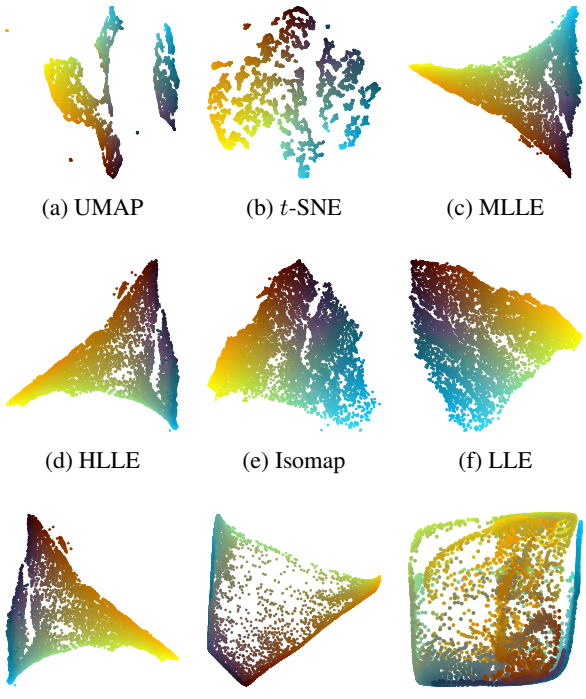
### Visualization of Surface Representations

Figures 2 – 5 plot the original surface and corresponding learned latent manifold embeddings for PBG-East, HZY-East, and HZY-West.




(a) PBG-East (b) HZY-West (c) HZY-East

Figure 2: Original surfaces.



(a) UMAP (b) $t$-SNE (c) MLLE

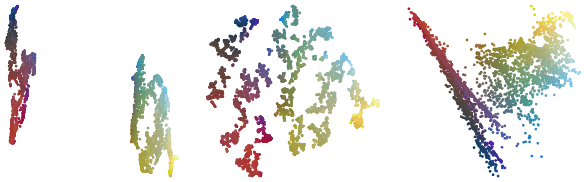
(d) HLLE (e) Isomap (f) LLE

(g) LTSA (h) WLLE (i) Dyland

Figure 3: Visualizations of manifolds on PBG-East.

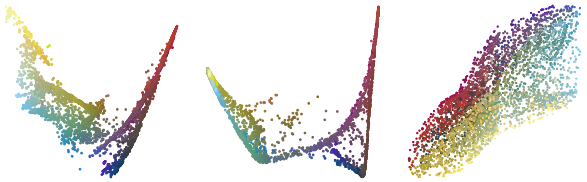
### Quantitative Comparisons on Manifold Learning

The quantitative results of comparisons between among manifold learning approaches for PBG-East and HZY-East are reported in Figure 6.



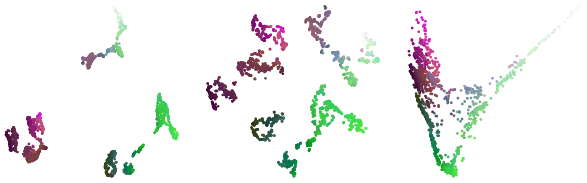
(a) UMAP (b) $t$-SNE (c) MLLE

(d) HLLE (e) Isomap (f) LLE



(g) LTSA (h) WLLE (i) Dyland

Figure 4: Visualizations of manifolds on HZY-West.



(a) UMAP (b) $t$-SNE (c) MLLE

(d) HLLE (e) Isomap (f) LLE

(g) LTSA (h) WLLE (i) Dyland

Figure 5: Visualizations of manifolds on HZY-East.

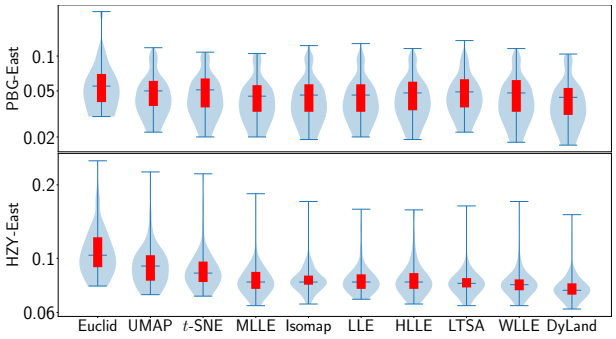



Figure 6: Quantitative comparisons between dynamic manifold and static manifold learning on PGB-East and HZY-East. ("Euclid" takes the original surface as input directly; red lines present the quartiles).

## Additional Ablation Studies

We represent the results of ablation studies of co-training on PBG-East and HZY-West in Table 1 and in Figure 7.

| | Model | RMSE | MAE | ACC | $R^2$ | EVS |
|---|---|---|---|---|---|---|
| **PBG-E** | DyLand-**S** | 0.053 | 0.041 | 0.710 | 0.401 | 0.412 |
| | DyLand-**W** | 0.023 | 0.018 | 0.965 | 0.452 | 0.489 |
| | DyLand | 0.016 | 0.013 | 0.978 | 0.487 | 0.496 |
| **HZY-E** | DyLand-**S** | 0.101 | 0.075 | 0.506 | 0.469 | 0.417 |
| | DyLand-**W** | 0.063 | 0.045 | 0.671 | 0.491 | 0.497 |
| | DyLand | 0.060 | 0.043 | 0.700 | 0.583 | 0.590 |

Table 1: Ablation results of DyLand on $W$ and $S$ co-training.

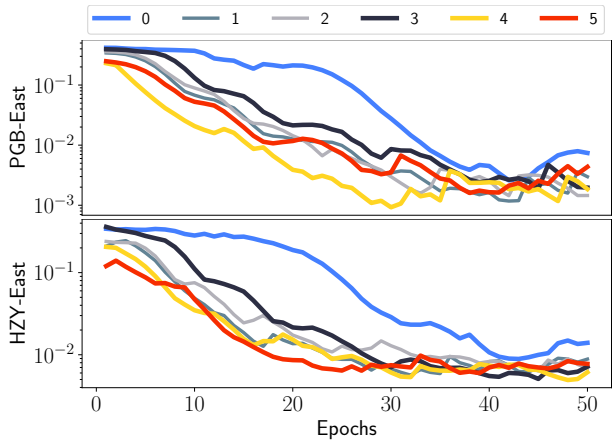
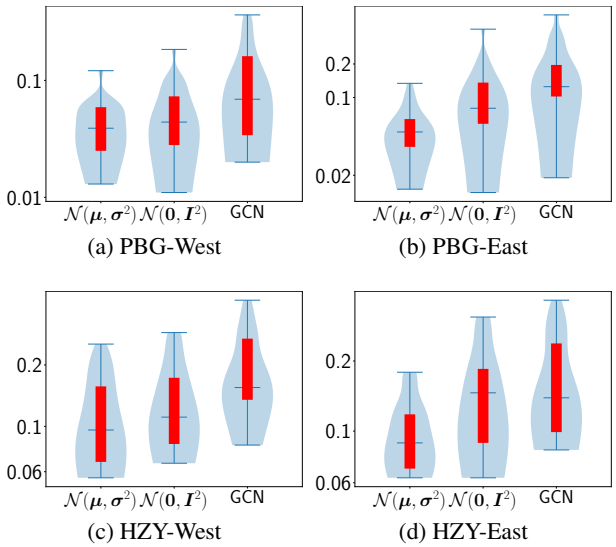



Figure 7: Training of DyLand in the first 50 epochs.

## Learning Spatio-Temporal Representations

We investigate the influence of different priors of $\mathbf{W}$, i.e., $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, $\mathcal{N}(\mathbf{0}, \boldsymbol{I}^2)$ and no prior (we replace the VGAE with a GCN). The results are reported in Figure 8, where the averaged standard deviation $\overline{\boldsymbol{\sigma}_{\mathbf{W}}} \simeq 1.2$ when approximating $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, and $\overline{\boldsymbol{\sigma}_{\mathbf{W}}} \simeq 0.9$ when approximating

$\mathcal{N}(\mathbf{0}, \boldsymbol{I}^2)$ – both of which are significantly smaller than the ground truth (cf. Figure 1). The mean $\overline{\boldsymbol{\mu}_{\mathbf{W}}} \in [-0.05, 0.05]$ in both cases. Compared to GCN, VGAE achieves better performance, which verifies the motivation to constrain the distribution of $\mathbf{W}$. We also find that, special designed prior is beneficial for model stability, but does not show significant superiority on optimal results, as observed by similar values of $\boldsymbol{\mu}_{\mathbf{W}}$ and larger values of $\boldsymbol{\sigma}_{\mathbf{W}}$. While encouraging $\mathbf{W}$ to follow a normal distribution with a larger standard deviation can relieve over-smoothing, the standard deviation is smaller than expected due to the over-smoothing and the under-fitting of VGAEs caused by fitting other parameters.




(a) PBG-West (b) PBG-East

(c) HZY-West (d) HZY-East

Figure 8: Influence of the prior of $\mathbf{W}$.

## References


Behrmann, J.; Grathwohl, W.; Chen, R. T. Q.; Duvenaud, D.; and Jacobsen, J.-H. 2019. Invertible Residual Networks. In *ICML*, 573–582.

Brehmer, J.; and Cranmer, K. 2020. Flows for simultaneous manifold learning and density estimation. In *NeurIPS*, 442–453.

Chen, C.; Li, C.; Chen, L.; Wang, W.; Pu, Y.; and Duke, L. C. 2018a. Continuous-Time Flows for Efficient Inference and Density Estimation. In *ICML*, 824–833.

Chen, R. T. Q.; Behrmann, J.; Duvenaud, D. K.; and Jacobsen, J.-H. 2019. Residual Flows for Invertible Generative Modeling. In *NeurIPS*, volume 32.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018b. Neural Ordinary Differential Equations. In *NeurIPS*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. arXiv:1605.08803.

Grathwohl, W.; Chen, R. T.; Bettencourt, J.; Sutskever, I.; and Duvenaud, D. 2018. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *ICLR*.

Hasanzadeh, A.; Hajiramezanali, E.; Narayanan, K.; Duffield, N.; Zhou, M.; and Qian, X. 2019. Semi-Implicit Graph Variational Auto-Encoders. In *NeurIPS*.

Hasenclever, L.; Tomczak, J. M.; van den Berg, R.; and Welling, M. 2017. Variational inference with orthogonal normalizing flows. In *Bayesian Deep Learning, NIPS 2017 workshop*.

Huang, C.-W.; Krueger, D.; Lacoste, A.; and Courville, A. 2018. Neural Autoregressive Flows. In *ICML*, 2078–2087.

Hutchinson, M. F. 1989. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3): 1059–1076.

Jaini, P.; Selby, K. A.; and Yu, Y. 2019. Sum-of-Squares Polynomial Flow. In *ICML*, 3009–3018.

Kim, H.; Lee, H.; Kang, W. H.; Lee, J. Y.; and Kim, N. S. 2020. SoftFlow: Probabilistic Framework for Normalizing Flow on Manifolds. In *NeurIPS*, 16388–16397.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative Flow with Invertible 1x1 Convolutions. *NeurIPS*, 31: 10215–10224.

Kingma, D. P.; and Welling, M. 2014. Auto-encoding variational bayes. In *ICLR*.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. arXiv:1611.07308.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Kong, Z.; and Chaudhuri, K. 2020. The Expressive Power of a Class of Normalizing Flow Models. In *AISTATS*, 3599–3609.

Kuznetsov, M.; and Polykovskiy, D. 2021. MolGrow: A Graph Normalizing Flow for Hierarchical Molecular Generation. In *AAAI*, 8226–8234.

Rezende, D.; and Mohamed, S. 2015. Variational Inference with Normalizing Flows. In *ICML*, 1530–1538.

Rezende, D. J.; Papamakarios, G.; Racaniere, S.; Albergo, M.; Kanwar, G.; Shanahan, P.; and Cranmer, K. 2020. Normalizing Flows on Tori and Spheres. In *ICML*, 8083–8092.

Tomczak, J. M.; and Welling, M. 2017. Improving Variational Auto-Encoders using Householder Flow. arXiv:1611.09630.

Wang, P. Z.; and Wang, W. Y. 2019. Riemannian Normalizing Flow on Variational Wasserstein Autoencoder for Text Modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 284–294.

Yin, M.; and Zhou, M. 2018. Semi-Implicit Variational Inference. In *ICML*, 5660–5669.