Journal of Machine Learning Research 25 (2024) 1-38 Submitted 6/23; Revised 8/24; Published 8/24

FineMorphs: Affine-Diffeomorphic Sequences for Regression

Michele Lohr MBIERBA2@JHU.EDU
Laurent Younes LAURENT.YOUNES@JHU.EDU
Department of Applied Mathematics and Statistics
Center for Imaging Science
The Johns Hopkins University
Baltimore, MD 21218-2683, USA

Editor: Jean-Philippe Vert

Abstract

A multivariate regression model of affine and diffeomorphic transformation sequences—FineMorphs—is presented. Leveraging concepts from shape analysis, model states are optimally "reshaped" by diffeomorphisms generated by smooth vector fields during learning. Affine transformations and vector fields are optimized within an optimal control setting, and the model can naturally reduce (or increase) dimensionality and adapt to large data sets via sub-optimal vector fields. An existence proof of solution and necessary conditions for optimality for the model are derived. Experimental results on real data sets from the UCI repository are presented, with favorable results in comparison with state-of-the-art in the literature, neural ordinary differential equation models, and densely-connected neural networks in TensorFlow.

Keywords: affine transformations, diffeomorphisms, machine learning, optimal control, regression, reproducing kernel Hilbert spaces, shape analysis

1. Introduction

We present FineMorphs—an affine-diffeomorphic sequence model for multivariate regression. Our approach combines arbitrary sequences of affine and diffeomorphic transformations with a training algorithm using concepts from optimal control. Predictors, estimated responses, and states in between are transformed or "reshaped" via diffeomorphisms of their respective ambient spaces, in an optimal way to facilitate learning.

Recall that diffeomorphisms of an open subset M of a Euclidean space $\mathbb{R}^d$ (where we will typically take $M = \mathbb{R}^d$) are one-to-one, invertible, C^1 transformations mapping M onto itself that have C^1 inverse. (If C^1 is replaced by C^0, one speaks of homeomorphisms.) Because diffeomorphisms form a group, arbitrary large deformations can be generated via the composition of many small ones, making them natural objects to utilize within a feed forward setting. In the limit of infinite compositions of transformations that differ infinitesimally from the identity, one finds the classical representation of diffeomorphisms as flows associated to ordinary differential equations (ODEs).

Several papers have recently explored the possibility of using homeomorphic or diffeomorphic transformations within feed-forward machine learning models. Discrete invertible versions of the ResNet architecture (He et al., 2016) were proposed as "normalizing flows" in Rezende and Mohamed (2015) (see Kobyzev et al. (2021) for a recent review), and extended to a time-continuous form in Chen et al. (2018); Rousseau et al. (2019); Dupont et al. (2019); Queiruga et al. (2020); with a recent survey in Ruthotto (2024). Continuous-time optimal control as a learning principle was proposed in Weinan (2017); Ganaba (2021); Owhadi (2023). Applications of deep residual neural networks (NNs) to the large deformation diffeomorphic metric mapping (LDDMM) framework of shape analysis have recently been explored (Amor et al., 2023; Wu and Zhang, 2023) as well as sub-Riemannian landmark matching as time-continuous NNs (Jansson and Modin, 2022).

A direct formalization of the diffeomorphic learning approach was proposed in Younes (2020) for classification by a single diffeomorphic layer sequence generated by optimal vector fields. While most flow-based learning approaches build dynamical systems that are adapted to NN implementations, diffeomorphic learning is presented as a non-parametric penalized learning problem, parametrized by a diffeomorphism of the data space. The penalty is specified as a Riemannian metric on the diffeomorphism group in a framework directly inspired from shape analysis (Younes, 2010). This allows for a smooth invertible reshaping of the underlying shape or manifold of a data set with explicit control of the smoothness of this transformation, in contrast with other methods including NNs. When applied to finite training data, the method reduces to a finite-, albeit large-, dimensional problem involving reproducing kernels (see Section 6). Shape analysis methods were also introduced for dimensionality reduction in Walder and Schölkopf (2009). Similar models were used combined with a shooting formulation for the comparison of geodesics in Vialard et al. (2020).

In this paper, we consider vector regression predictors of the form

$$x \in \mathbb{R}^{d_X} \mapsto A_m \circ \varphi_m \circ A_{m-1} \circ \cdots \circ \varphi_1 \circ A_0(x) \in \mathbb{R}^{d_Y}, \tag{1}$$

where A_q, $q = 0, \ldots, m$, are affine transformations from $\mathbb{R}^{d_q}$ to $\mathbb{R}^{d_{q+1}}$, and φ_q, $q = 1, \ldots, m$, are diffeomorphisms on $\mathbb{R}^{d_q}$. In this setting, a d_Y-dimensional output variable is predicted by the transformation of a d_X-dimensional input through an arbitrary number and order of arbitrary affine and diffeomorphic transformations. In contrast with the single diffeomorphic layer sequence model in Younes (2020), these arbitrary affine-diffeomorphic sequences allow for greater model complexity and provide a natural framework for automated data scaling and dimensionality reduction. Additionally, we extend the diffeomorphic learning model to more general sub-optimal vector fields parametrized by control points (see Section 8), enabling training on very large data sets. Combined with a GPU implementation, this allows for experiments on data sets beyond smaller-sized, simulated data sets to real-world data with larger, more realistic dimensions and sizes. Finally, we provide an existence proof of solution to the variational problem and a derivation of necessary optimality conditions. The existence proof was loosely sketched in Younes (2020), and is provided here with full details and extended to the more general situation addressed in the present paper.

We test our diffeomorphic regression models on real data sets from the UCI repository (Dua and Graff, 2017), with favorable results in comparison with the literature and with neural ODEs (NODEs) (Chen et al., 2018; Dupont et al., 2019) and densely-connected NNs (DNNs) in TensorFlow (Abadi et al., 2015). We note improved performance with multiple sequential diffeomorphic modules with decreasing kernel sizes as well as a robustness of our models to "out-of-distribution" testing. For the largest data set in our experiments, in both dimensionality and number size, our model reduces dimensionality through affine transformations and reduces computational complexity through the control points method, with a tractable run time and good predictive results in comparison with the literature and DNNs.

2. Notation

For our multivariate regression setting, $X : \Omega \to \mathbb{R}^{d_X}$ is the predictor variable and $Y : \Omega \to \mathbb{R}^{d_Y}$ is the response. The training data set is denoted $\mathcal{T}_0 = (x_1, y_1, \ldots, x_n, y_n)$. The training predictors are $\boldsymbol{x} = (x_1, \ldots, x_n) \in (\mathbb{R}^{d_X})^n$ and training responses are $\boldsymbol{y} = (y_1, \ldots, y_n) \in (\mathbb{R}^{d_Y})^n$. We define the operator $\iota_j : \mathbb{R}^d \to \mathbb{R}^{d+j}$, where $\iota_j(x)$ appends j zero coordinates to x, and the operator $\pi_j : \mathbb{R}^d \to \mathbb{R}^{d-j}$, where $\pi_j(x)$ removes the last j coordinates from x. For matrix notation, if k, l are two integers, $\mathcal{M}_{k,l}(\mathbb{R})$ is the space of all $k \times l$ real matrices, reducing to $\mathcal{M}_k(\mathbb{R})$ for square $k \times k$ real matrices. The $d \times d$ identity matrix is denoted I_d. When applied to vectors and matrices, the norm $\|\cdot\|$ is the Euclidean and Frobenius norm, respectively. For time-dependent vector fields

$$\begin{aligned} v : \mathbb{R} \times \mathbb{R}^d &\to \mathbb{R}^d \\ (t, x) &\mapsto v(t, x), \end{aligned}$$

we will denote by v the mapping $t \mapsto v(t)$, where $v(t)$ is the time-indexed vector field $x \mapsto v(t, x)$. In particular, the time-dependent vector field v in the Bochner space $L^2(I, V)$ will represent the mapping $t \in I \mapsto v(t) \in V$, where V is a Hilbert space.

3. Model

We consider the following regression model approximating Y by $f(X)$, in which we complete (1) by possibly padding zeros in input and removing coordinates in output,

$$f : x \in \mathbb{R}^{d_X} \mapsto \pi_r \left(A_m \circ \varphi_m \circ A_{m-1} \circ \cdots \circ \varphi_1 \circ A_0(\iota_s(x)) \right) \in \mathbb{R}^{d_Y}.$$

Here, ι_s pads the input with s zeros so that $d_0 = d_X + s$, and π_r removes the last r coordinates from the model output so that $d_{m+1} = d_Y + r$. Advantages of adding "dummy" dimensions are discussed in Section 9. In contrast to the single affine layered approach of standard linear regression, this model alternates $m+1$ affine transformations and m diffeomorphic layers, denoted as A and D modules, respectively, starting and ending with affine modules. For affine modules A_q, $q = 0, \ldots, m$, the corresponding affine transformations are

$$A_q : x \in \mathbb{R}^{d_q} \mapsto M_q x + b_q \in \mathbb{R}^{d_{q+1}},$$

where $M_q \in \mathcal{M}_{d_{q+1}, d_q}(\mathbb{R})$, $b_q \in \mathbb{R}^{d_{q+1}}$. For diffeomorphic modules D_q, $q = 1, \ldots, m$, the corresponding diffeomorphisms and their domains are φ_q and $\mathbb{R}^{d_q}$, respectively.

The values of s and r, and the internal dimensions $d_1, \ldots, d_m$ are parts of the design of the model, i.e., they are user-specified. Given them, the dimensions of the linear operators are uniquely determined, and so are the spaces on which the diffeomorphisms operate. Any module in a sequence with identical input and output dimensions can be set to the identity map, id, which allows for simple definitions of submodels from an initial sequence of modules (obviously, one wants to keep at least one A module and at least one D module

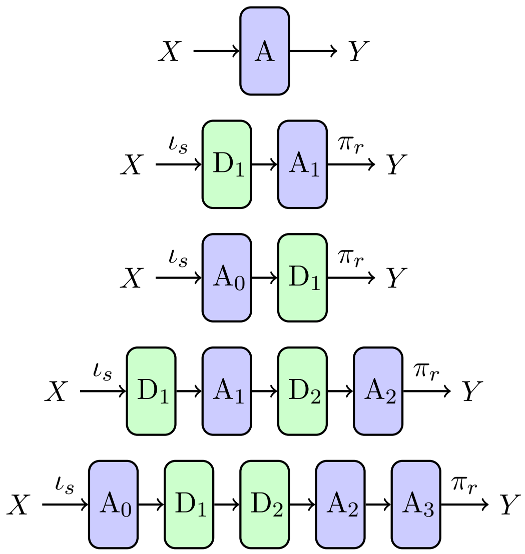

Figure 1: Standard linear regression (top) followed by four example transformation sequences that can be operated by the FineMorphs model, with naming convention (from top to bottom): A, DA, AD, DADA or (DA)^2, and ADDAA or AD^{2}A^2 (A: affine module; D: diffeomorphic module). Identity modules are omitted.

free to optimize by the system). The flexibility of assigning module dimensions as well as arbitrary modules to the identity generalizes our model from a simple and fixed alternating sequence to an arbitrary sequence of arbitrary affine and diffeomorphic transformations. In this setting, affine modules can provide not only useful data scaling prior to diffeomorphic transforms but also a natural approach to dimensionality reduction or increase. In the following, the naming convention for sequences includes only non-identity modules, e.g., the sequence of modules A$_0$, D$_1$, A$_1$, D$_2$, A$_2$, D$_3$, and A$_3$, where A$_1$ and D$_3$ are identities, is denoted ADDAA. For sequence names containing repetitive module or module subsequence elements, we further adopt a simplified notation superscripting the repetition, e.g., ADDAA can be expressed as AD^{2}A^2, and sequence ADAD$\cdots$A with x sequential AD module pairs before the final A can be denoted as (AD)$^{\mathrm{x}}$A. Several sequence examples are illustrated in Figure 1, including the smallest possible sequences that can be represented in our model, DA and AD.

4. Objective Function

Learning is implemented by minimizing the objective function

$$\sum_{q=1}^{m} d_{V_q}(\mathrm{id}, \varphi_q)^2 + \lambda \sum_{q=0}^{m} U_q(A_q) + \frac{1}{\sigma^2} \sum_{k=1}^{n} \Gamma_k(\pi_r(A_m \circ \varphi_m \circ A_{m-1} \circ \cdots \circ \varphi_1 \circ A_0(\iota_s(x_k))))$$

over $\varphi_1, \ldots, \varphi_m, A_0, \ldots, A_m$. The objective function combines an optimal deformation cost d_{V_q}, an affine cost U_q, and a standard loss function or endpoint cost Γ_k. In our setting, d_{V_q}

is a Riemannian distance in a group of diffeomorphisms of $\mathbb{R}^{d_q}$ described in Section 5, U_q is a ridge regularization function

$$U_q(A) = \|M\|^2 = \text{trace}(M^\top M)$$

for affine transformations $A : \mathbb{R}^{d_q} \to \mathbb{R}^{d_{q+1}}$ of the form $A(x) = Mx + b$, $M \in \mathcal{M}_{d_{q+1},d_q}(\mathbb{R})$, $b \in \mathbb{R}^{d_{q+1}}$, and Γ_k is a squared error loss function

$$\Gamma_k(\cdot) = \|y_k - (\cdot)\|^2$$

for comparison of experimental responses with model predictions.

5. Distance over Diffeomorphisms

Spaces of diffeomorphisms are defined as follows (Beg et al., 2005; Miller et al., 2015; Younes, 2010). Let $\mathbf{B}_p = C_0^p(\mathbb{R}^d, \mathbb{R}^d)$ denote the space of C^p vector fields on $\mathbb{R}^d$ that tend to zero (together with their first p derivatives) at infinity. This is a Banach space for the norm

$$\|f\|_{p,\infty} = \max_{0\le k\le p} \left\| d^k f \right\|_\infty ,$$

where $\|\cdot\|_\infty$ denotes the usual supremum norm. Let V denote a Hilbert space of vector fields on $\mathbb{R}^d$, continuously embedded in $\mathbf{B}_p$ for some $p \geq 1$, so that there exists a $C > 0$ such that

$$\|f\|_{p,\infty} \le C \|f\|_V , \tag{2}$$

for all $f \in V$, where $\|\cdot\|_V$ is the Hilbert norm on V with inner product $\langle\cdot,\cdot\rangle_V$.

Diffeomorphisms can be generated as flows of ODEs associated with time-dependent elements of V. Let $\mathcal{H}$ denote the Hilbert space $L^2([0,1], V)$ of time-dependent vector fields, so that $v \in \mathcal{H}$, if and only if $v(t) \in V$ for $t \in [0,1]$, v is measurable, and

$$\|v\|_{\mathcal{H}}^2 = \int_0^1 \|v(t)\|_V^2 dt < \infty,$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the norm on $\mathcal{H}$ with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$. Then the ODE

$$\partial_t y(t) = v(t)(y(t))$$

has a unique solution over $[0,1]$ given any initial condition $y(0) = x$. The flow of the ODE is the function

$$\boldsymbol{\varphi}_v : (t, x) \mapsto y(t),$$

where $y(t)$ is the solution starting at x, after t units of time. This function is the unique flow of $\mathbb{R}^d$-diffeomorphisms satisfying the dynamical system

$$\begin{aligned} \partial_t \boldsymbol{\varphi}_v(t,x) &= v(t)(\boldsymbol{\varphi}_v(t,x)) \\ \boldsymbol{\varphi}_v(0,x) &= x \end{aligned}$$

over $t \in [0,1]$. We will often write $\boldsymbol{\varphi}_v(t)$ for the time-indexed function $x \mapsto \boldsymbol{\varphi}_v(t,x)$ satisfying

$$\begin{aligned}\partial_t \boldsymbol{\varphi}_v(t) &= v(t) \circ \boldsymbol{\varphi}_v(t), \quad t \in [0,1] \\ \boldsymbol{\varphi}_v(0) &= \mathrm{id}.\end{aligned}$$

The set of diffeomorphisms that can be generated in such a way forms a group denoted Diff_V, such that a flow path associated with some $v \in \mathcal{H}$ is a curve on Diff_V. Let $\frac{1}{2}\|v(t)\|_V^2$ denote the kinetic energy associated with the flow's velocity at time t along this curve. Given $\psi \in \mathrm{Diff}_V$, we define the optimal deformation cost from id to ψ as the minimal kinetic energy among all curves between id and ψ on Diff_V, i.e., the minimum of $\int_0^1 \|v(t)\|_V^2 dt$ over all $v \in \mathcal{H}$ such that $\boldsymbol{\varphi}_v(1) = \psi$. A distance $d_V(\cdot,\cdot)$ can then be defined on Diff_V with the right-invariance property $d_V(\psi, \tilde{\psi}) = d_V(\psi \circ \psi', \tilde{\psi} \circ \psi')$ for $\psi, \tilde{\psi}, \psi' \in \mathrm{Diff}_V$. Given $\psi, \psi' \in \mathrm{Diff}_V$, $d_V(\psi, \psi') = d_V(\mathrm{id}, \psi' \circ \psi^{-1})$ and

$$d_V(\mathrm{id}, \psi)^2 = \min_{v \in \mathcal{H}} \left\{ \int_0^1 \|v(t)\|_V^2 dt : \boldsymbol{\varphi}_v(1) = \psi \right\} .$$

In our setting of m distinct D modules, we assume for each D_q module the corresponding Hilbert space V_q of vector fields on $\mathbb{R}^{d_q}$ and Hilbert space $L^2([0,1], V_q)$ denoted $\mathcal{H}_q$, and let the time-dependent vector fields $v_q \in \mathcal{H}_q$ generate the corresponding Diff_{V_q} space of diffeomorphisms. Our optimal deformation cost can then be expressed in terms of the vector fields as

$$\sum_{q=1}^m d_{V_q}(\mathrm{id}, \varphi_q)^2 = \sum_{q=1}^m \min_{v_q \in \mathcal{H}_q} \left\{ \int_0^1 \|v_q(t)\|_{V_q}^2 dt : \boldsymbol{\varphi}_{v_q}(1) = \varphi_q \right\}.$$

We introduce forward states between modules as $\xi^0, \zeta^1, \xi^1, \zeta^2, \ldots, \xi^m, \zeta^{m+1}$, as shown in Figure 2, given by

$$\begin{cases} \zeta_k^{q+1} = A_q(\xi_k^q) \\ \xi_k^q = \boldsymbol{\varphi}_{v_q}(1)(\zeta_k^q) \end{cases} \quad k = 1, \ldots, n$$

for outputs of corresponding A_q and D_q modules, respectively, with $\xi_k^0 = \iota_s(x_k)$, ensuring that

$$\zeta_k^{m+1} = A_m \circ \boldsymbol{\varphi}_{v_m}(1) \circ A_{m-1} \circ \cdots \circ \boldsymbol{\varphi}_{v_1}(1) \circ A_0(\iota_s(x_k)).$$

The objective function becomes

$$\sum_{q=1}^m \int_0^1 \|v_q(t)\|_{V_q}^2 dt + \lambda \sum_{q=0}^m U_q(A_q) + \frac{1}{\sigma^2} \sum_{k=1}^n \Gamma_k(\pi_r(\zeta_k^{m+1})) \tag{3}$$

minimized over $A_0, \ldots, A_m$, and $v_q \in \mathcal{H}_q$, $q = 1, \ldots, m$, such that $\boldsymbol{\varphi}_{v_q}(t)$ satisfies

$$\begin{aligned}\partial_t \boldsymbol{\varphi}_{v_q}(t) &= v_q(t) \circ \boldsymbol{\varphi}_{v_q}(t), \quad t \in [0,1] \\ \boldsymbol{\varphi}_{v_q}(0) &= \mathrm{id}.\end{aligned}$$

When the Hilbert norms on V_q are translation invariant, a minimizer of this objective function always exists. This is demonstrated in Appendix A.

6. Kernel Reduction

The embedding assumption in equation (2) implies that $V_1, \dots, V_m$ are vector-valued reproducing kernel Hilbert spaces (RKHSs) (Aronszajn, 1950; Wahba, 1990; Joshi and Miller, 2000; Miller et al., 2002; Vaillant et al., 2004; Micchelli and Pontil, 2005). By Riesz's representation theorem, each V_q has an associated matrix-valued kernel function

$$K_q : \mathbb{R}^{d_q} \times \mathbb{R}^{d_q} \to \mathcal{M}_{d_q}(\mathbb{R})$$

that reproduces every function in V_q. More precisely, for every $y, a \in \mathbb{R}^{d_q}$, there exists a unique element $K_q(\cdot, y)a$ of V_q such that

$$K_q(\cdot, y)a : x \in \mathbb{R}^{d_q} \mapsto K_q(x, y)a$$

and

$$\langle K_q(\cdot, y)a, f \rangle_{V_q} = a^\top f(y)$$

for all $f \in V_q$. These properties imply

$$\langle K_q(\cdot, x)a, K_q(\cdot, y)b \rangle_{V_q} = a^\top K_q(x, y)b$$

and thus symmetry, $K_q(y, x) = K_q(x, y)^\top$, and positive semi-definiteness for all $x, y, a, b \in \mathbb{R}^{d_q}$. Conversely, by the Moore–Aronszajn theorem, any matrix-valued kernel that is symmetric and positive semi-definite induces the corresponding vector-valued RKHS of functions reproducible by this kernel.

An RKHS argument similar to the kernel trick used in standard kernel methods can reduce the dimension of our problem as follows. Let

$$z_k^q(t) = \boldsymbol{\varphi}_{v_q}(t)(\zeta_k^q), \quad k = 1, \dots, n$$

represent the time-dependent states in $\mathbb{R}^{d_q}$ of module D_q, and denote the array of n states as $\boldsymbol{z}^q(\cdot) = (z_1^q(\cdot), \dots, z_n^q(\cdot))$. The dependence of our endpoint cost on each vector field $v_q(t)$ is through the n trajectories

$$\partial_t z_k^q(t) = v_q(t)(z_k^q(t))$$

generating the n corresponding endpoints $\zeta_1^{m+1}, \dots, \zeta_n^{m+1}$. The vector fields minimizing this cost are regularized by the RKHS norm $\| \cdot \|_{V_q}$ on their respective spaces V_q. By the representer theorem, these minimizers must then take the form

$$v_q(t)(\cdot) = \sum_{l=1}^{n} K_q(\cdot, z_l^q(t)) a_l^q(t),$$

where $\boldsymbol{a}^q(\cdot) = (a_1^q(\cdot), \dots, a_n^q(\cdot))$ are the unknown time-dependent vectors in $\mathbb{R}^{d_q}$ to be determined. In this reduced representation, our objective function

$$\sum_{q=1}^{m} \int_0^1 \sum_{k,l=1}^{n} a_k^q(t)^\top K_q(z_k^q(t), z_l^q(t)) a_l^q(t) dt + \lambda \sum_{q=0}^{m} U_q(A_q) + \frac{1}{\sigma^2} \sum_{k=1}^{n} \Gamma_k(\pi_r(\zeta_k^{m+1}))$$

is minimized over $\boldsymbol{a}^1(\cdot), \ldots, \boldsymbol{a}^m(\cdot)$, $A_0, \ldots, A_m$, subject to the system of trajectories and initial conditions

$$\begin{aligned} \partial_t z_k^q(t) &= \sum_{l=1}^n K_q(z_k^q(t), z_l^q(t)) a_l^q(t) \\ z_k^q(0) &= \zeta_k^q = A_{q-1}(\xi_k^{q-1}) \\ \xi_k^q &= z_k^q(1) \end{aligned}$$

and initialization $\xi_k^0 = \iota_s(x_k)$. Our learning problem can now be solved as an optimal control problem with a finite dimensional control space.

7. Optimal Control

An optimal control steers the state of a system from a given initial state to a final state while optimizing an objective function, typically a running cost and an endpoint cost to be minimized. Our learning problem can be solved in an optimal control framework, as we seek the optimal deformations (control) and affine parameters for our system of trajectories and initial conditions such that a deformation (running) cost and a learning (endpoint) cost are minimized.

Assuming existence of solutions, the Pontryagin Maximum Principle (PMP) (Hocking, 1991; Macki and Strauss, 2012) provides necessary conditions for optimality in optimal control settings. By the PMP, an optimal control and trajectory must also solve a Hamiltonian system with a corresponding costate and a stationarity condition. We derive the PMP for our model within the Lagrangian variational framework in Appendix B, with the resulting solutions as follows.

First define backpropagation states between modules as $\rho^1, \eta^1, \rho^2, \ldots, \eta^m, \rho^{m+1}$, as shown in Figure 2, where

$$\begin{cases} \eta_k^q = M_q^\top \rho_k^{q+1} \\ \rho_k^q = \mathcal{F}_q(\eta_k^q) \end{cases} \qquad k = 1, \ldots, n$$

are states propagating back from corresponding A_q and D_q modules, respectively, with

$$\rho_k^{m+1} = -\tfrac{1}{\sigma^2} \iota_r(\nabla \Gamma_k(\pi_r(\zeta_k^{m+1}))).$$

$\mathcal{F}_q(\eta_k^q)$ is obtained by solving the ODEs

$$\begin{cases} \partial_t z_k^q(t) = \sum\limits_{l=1}^n K_q(z_k^q(t), z_l^q(t)) a_l^q(t), \quad z_k^q(0) = \zeta_k^q \\ \partial_t p_k^q(t) = -\sum\limits_{l=1}^n \nabla_1 K_q(z_k^q(t), z_l^q(t))(p_k^q(t)^\top a_l^q(t) + a_k^q(t)^\top p_l^q(t) - 2a_k^q(t)^\top a_l^q(t)), \quad p_k^q(1) = \eta_k^q \end{cases}$$

and assigning $\mathcal{F}_q(\eta_k^q) = p_k^q(0)$, where $p_k^q(t)$ are the time-dependent costates in $\mathbb{R}^{d_q}$ of module D_q, with array of n costates denoted $\boldsymbol{p}^q(\cdot) = (p_1^q(\cdot), \ldots, p_n^q(\cdot))$. Note the states $z_k^q(t)$ are calculated on the forward pass through the model and cached for the backpropagation pass.

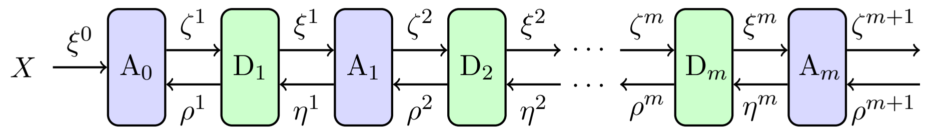

Figure 2: General FineMorphs model of alternating A and D modules. Forward states are $\xi^0, \zeta^1, \xi^1, \zeta^2, \ldots, \xi^m, \zeta^{m+1}$, with $\xi_k^q = \boldsymbol{\varphi}_{v_q}(1)(\zeta_k^q)$, $\zeta_k^{q+1} = A_q(\xi_k^q)$, and $\xi_k^0 = \iota_s(x_k)$, for data point index k. Backpropagation states are $\rho^1, \eta^1, \rho^2, \ldots, \eta^m, \rho^{m+1}$, with $\eta_k^q = M_q^\top \rho_k^{q+1}$, $\rho_k^q = \mathcal{F}_q(\eta_k^q)$, and $\rho_k^{m+1} = -\frac{1}{\sigma^2}\iota_r(\nabla\Gamma_k(\pi_r(\zeta_k^{m+1})))$.

Let G denote our objective function. The gradients for determining our optimal control parameters $\boldsymbol{a}^1(\cdot), \ldots, \boldsymbol{a}^m(\cdot)$ and affine parameters $A_0, \ldots, A_m$ are then

$$\partial_{a_k^q(t)} G = \sum_{l=1}^{n} K_q(z_k^q(t), z_l^q(t))(2a_l^q(t) - p_l^q(t)), \quad k = 1, \ldots, n, \quad q = 1, \ldots, m$$

$$\partial_{M_q} G = \lambda \partial_{M_q} U_q(A_q) - \sum_{k=1}^{n} \rho_k^{q+1} \xi_k^{q\top}, \quad q = 0, \ldots, m$$

$$\partial_{b_q} G = -\sum_{k=1}^{n} \rho_k^{q+1}, \quad q = 0, \ldots, m,$$

which can be used in gradient descent methods as the directions in which to step the current parameters to minimize the objective function. Once the parameters are updated, another forward pass through our model is run, recalculating the forward states and objective function, followed by backpropagation, recalculating the backpropagation states and gradients. The parameters are then updated again, and the cycle repeated, until a sufficient minimum in the objective function or total gradient is achieved.

8. Control Points Method

Large data sets and models are time and resource prohibitive in many machine learning tasks. Our model can be naturally adapted to large data sets by approximating the optimal vector fields with more general "sub-Riemannian" or sub-optimal ones parametrized by a finite set of arbitrary points in the data space called control points. During learning, the corresponding lower-complexity diffeomorphisms are applied to the entire training data set for analysis in the endpoint cost. Similar approximations were introduced in shape analysis (see Younes et al. (2020) for a review and references) and in Walder and Schölkopf (2009); Vialard et al. (2020).

To approximate the optimal vector fields, we choose a set of points from the data space of size $n_S \leq n$, and, without loss of generality, constrain these points to the training data set. Without further loss of generality, we renumber the training data such that its first n_S elements coincide with this subset. Then the sub-optimal vector fields are defined as

$$v_q(t)(\cdot) = \sum_{l=1}^{n_S} K_q(\cdot, z_l^q(t)) a_l^q(t),$$

where $(z_1^q(\cdot), \ldots, z_{n_S}^q(\cdot))$ and $\boldsymbol{a}^q(\cdot) = (a_1^q(\cdot), \ldots, a_{n_S}^q(\cdot))$ are the states corresponding to this subset and the control parameters, respectively. The resulting objective function

$$\sum_{q=1}^{m} \int_0^1 \sum_{k,l=1}^{n_S} a_k^q(t)^\top K_q(z_k^q(t), z_l^q(t)) a_l^q(t) dt + \lambda \sum_{q=0}^{m} U_q(A_q) + \frac{1}{\sigma^2} \sum_{k=1}^{n} \Gamma_k(\pi_r(\zeta_k^{m+1})) \tag{4}$$

is minimized over $\boldsymbol{a}^1(\cdot), \ldots, \boldsymbol{a}^m(\cdot)$, $A_0, \ldots, A_m$, subject to the system of trajectories

$$\partial_t z_k^q(t) = \sum_{l=1}^{n_S} K_q(z_k^q(t), z_l^q(t)) a_l^q(t)$$

with the same initial conditions and initialization as in the optimal vector fields case. The existence of a minimizer of this objective function is demonstrated in Appendix A. The PMP is derived in Appendix B, where the more general expressions for the costate trajectories and gradients for the optimal control for this case are found.

9. Dummy Dimensions

Adding "dummy" dimensions to a data set can provide a benefit in our setting (Younes, 2020; Dupont et al., 2019). In cases where a diffeomorphism of the given domain cannot reshape the data to within an affine transformation of the true responses for successful regression—or is too costly to do so—adding dimensions can provide a more viable or less costly pathway for the diffeomorphism. An example is illustrated with the two-dimensional Rings on the left in Figure 3, where the data point locations and colors represent the predictors and true responses, respectively. Zero padding the predictors with one additional dimension then applying our baseline model (described in Section 14) leads to a linear representation of the true responses by a simple diffeomorphism of the predictors as shown on the right. To facilitate the use of the extra dimensions by the diffeomorphisms, these dimensions can be initialized with random number values small enough to break the data symmetry without impacting the data structure.

10. Implementation

The model is implemented in Python using an Euler discretization approach, with objective function

$$\sum_{q=1}^{m} \frac{1}{T_q} \sum_{i=0}^{T_q-1} \sum_{k,l=1}^{n} a_k^q(\tfrac{i}{T_q})^\top K_q(z_k^q(\tfrac{i}{T_q}), z_l^q(\tfrac{i}{T_q})) a_l^q(\tfrac{i}{T_q}) + \lambda \sum_{q=0}^{m} \|M_q\|^2 + \frac{1}{\sigma^2} \sum_{k=1}^{n} \Gamma_k(\pi_r(\zeta_k^{m+1}))$$

minimized over

(i) $\boldsymbol{a}^q(\tfrac{i}{T_q}) = (a_1^q(\tfrac{i}{T_q}), \ldots, a_n^q(\tfrac{i}{T_q})) \in (\mathbb{R}^{d_q})^n, \quad i = 0, \ldots, T_q - 1, \quad q = 1, \ldots, m$

(ii) $M_q \in \mathcal{M}_{d_{q+1}, d_q}(\mathbb{R}),\ b_q \in \mathbb{R}^{d_{q+1}}, \quad q = 0, \ldots, m,$

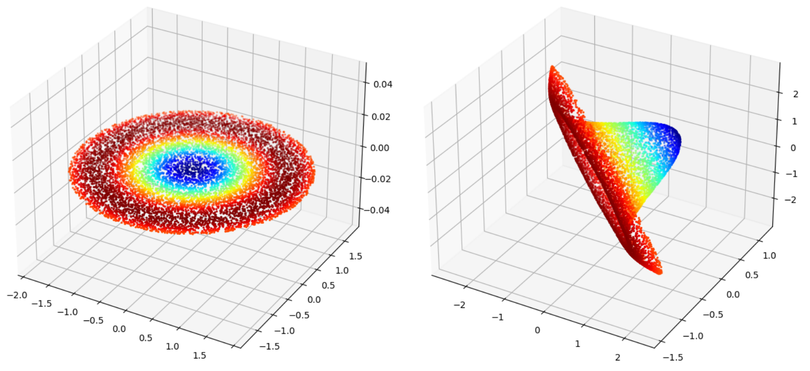

Figure 3: Two-dimensional Rings data set (left) with a linear representation of the color-coded true responses (right) following a diffeomorphism on the domain with one added dummy dimension.

subject to

$$z_k^q(\tfrac{i+1}{T_q}) = z_k^q(\tfrac{i}{T_q}) + \frac{1}{T_q}\sum_{l=1}^{n} K_q(z_k^q(\tfrac{i}{T_q}), z_l^q(\tfrac{i}{T_q}))a_l^q(\tfrac{i}{T_q})$$

with $z_k^q(0) = \zeta_k^q = M_{q-1}\xi_k^{q-1} + b_{q-1}$, $\xi_k^q = z_k^q(1)$, and initialization $\xi_k^0 = \iota_s(x_k)$. The model parameters are initialized as

(i) $\boldsymbol{a}^q(\tfrac{i}{T_q}) = 0 \in (\mathbb{R}^{d_q})^n, \quad i = 0, \ldots, T_q - 1, \quad q = 1, \ldots, m$

(ii) $M_q \sim \mathcal{N}(0, \tfrac{1}{d_q}) \in \mathcal{M}_{d_{q+1},d_q}(\mathbb{R})$, $b_q = 0 \in \mathbb{R}^{d_{q+1}}, \quad q = 0, \ldots, m.$

We include an option to speed up kernel computations using PyKeOps (Charlier et al., 2021) with user-specified precision and GPUs. Our optimization algorithms are gradient descent methods implemented with line search.

To run the model, the user specifies an arbitrary sequence and number of non-identity A and D modules, dimension parameters s, r, and $d_1, \ldots, d_m$, ridge regularization weight λ, and optimization algorithm parameters for gradient descent, including stopping thresholds and maximum number of iterations. For each D_q module, the user specifies the kernel type and the number of discretized time points T_q for state and costate propagation and the control variables. For each kernel K_q, the algorithm assigns a default kernel width h_q of 0.5, as the affine module preceding D_q automatically scales and adapts its input to the kernel width of the subsequent D_q, removing the need for user-provided kernel widths. Input and output dimension assignments for each module in the sequence are automated by our algorithm based on d_X, d_Y, s, r, and the inner module dimensions provided by the user. The normalization factor σ of the error term is determined by our model as a function of the noise in the training data, as described in Section 12.

11. Data Preprocessing

Prior to training, the $\boldsymbol{x}$ and $\boldsymbol{y}$ training data in $\mathcal{T}_0$ are standardized by the model to zero mean and unit variance by subtracting their respective means, $\mu_X \in \mathbb{R}^{d_X}$ and $\mu_Y \in \mathbb{R}^{d_Y}$, and dividing by their respective standard deviations, $\sigma_X \in \mathbb{R}^{d_X}$ and $\sigma_Y \in \mathbb{R}^{d_Y}$. For $s > 0$, s extra dimensions are then appended to the training predictors by n vector draws from $\mathcal{N}(0, 0.01^2) \in \mathbb{R}^s$. Prior to testing, the test predictors are standardized using the standardization parameters of the training predictors, μ_X and σ_X, and then appended with n_{test} zero vectors $0 \in \mathbb{R}^s$, where n_{test} is the number of data points in the test set.

12. Normalization Factor and Model Training

To determine an optimal penalty for endpoint matching errors and prevent overfitting to data noise, the σ normalization factor of the error term is calculated by the model as follows. First, the noise of the "data manifold," σ^2_{data}, is estimated, and a noise threshold, σ^2_{thresh}, is set to this value with a minimum cap to avoid overfitting in the case of low noise,

$$\sigma^2_{\text{thresh}} = \max\left\{\sigma^2_{\text{data}}, 0.01\right\}.$$

Training begins, with the normalization factor initialized to a scale of the noise threshold,

$$\sigma^2 = n^{1/2}\sigma^2_{\text{thresh}},$$

(the $\sqrt{n}$ factor was determined empirically as a good choice for initialization) then iteratively decreased after a fixed number of training steps until the training error falls below the noise threshold. Using the final value for σ, the training algorithm is then run until convergence.

13. Evaluation Metric

The diffeomorphisms and affine transformations learned on the training set are applied to the corresponding test set for performance analysis. Specifically, the test predictors are forward propagated through the model, transformed in turn by the learned affine transformations of each A_q and the vector fields of each D_q, the latter functions of the learned $\boldsymbol{a}^q(\cdot)$ and cached $\boldsymbol{z}^q(\cdot)$. The evaluation metric is the square root of the mean square error (MSE), or root-MSE (RMSE), between the model outputs $\zeta^{m+1}_{k,test}$ and the test experimental responses

$$\sqrt{\frac{1}{n_{test}}\sum_{k=1}^{n_{test}} \Gamma_k(\sigma_Y \odot \pi_r(\zeta^{m+1}_{k,test}) + \mu_Y)},$$

which we will denote test RMSE. Lower test RMSE signifies better performance.

14. Baseline Experiments

While DA and AD are the smallest possible sequences that can be represented in our model, the AD sequence is not as practical for regression purposes, and the DA sequence requires the user to specify a data-specific kernel width for the D_1 diffeomorphism. Therefore, we consider the ADA model, which is the sequence case for $m = 1$ and no identity modules, as

$$\begin{tikzcd} X \arrow[r, "\iota_1"] & \boxed{\mathrm{A}_0} \arrow[r] & \boxed{\mathrm{D}_1} \arrow[r] & \boxed{\mathrm{A}_1} \arrow[r, "\pi_0"] & Y \end{tikzcd}$$

Figure 4: ADA transformation sequence used in the baseline experiments.

our simplest regression model sequence, and we choose this baseline model for our experiments, as shown in Figure 4. Additionally, we choose the simplest reasonable values for the remaining hyperparameters of our model. We set $\lambda = 1$ and assign dimensions $s = 1$, $r = 0$, and $d_1 = d_X + s$, ensuring the dummy dimension added to the data set is carried through module A_0 to the diffeomorphism in D_1. For module D_1, we set $T_1 = 10$, and we construct a matrix-valued kernel from the scalar Matérn kernel and the identity matrix I_{d_1} (Younes, 2020). In particular,

$$K_1(x, y) = \left(1 + u + 0.4u^2 + \frac{1}{15}u^3\right) e^{-u}\mathrm{I}_{d_1}, \quad u = \frac{|y - x|}{h_1}$$

with default kernel width $h_1 = 0.5$. Kernel computations are performed using PyKeOps with an NVIDIA RTX A5000 GPU with CUDA 12.1. The optimization algorithm is the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) with Wolfe conditions on the line search. Early stopping, typically used to prevent overfitting, is avoided by setting the maximum number of gradient descent iterations large enough to ensure numerical convergence. The FineMorphs code and data sets used in the experiments are available at `https://github.com/diffeomorphic-learning/finemorphs`.

Our model is tested on nine UCI data sets—*Concrete*, *Energy*, *Kin8nm*, *Naval*, *Power*, *Protein*, *Wine Red* (denoted *Wine*), *Yacht*, and *Year*—with standard splits originally generated for the experiments in Hernández-Lobato and Adams (2015) and gap splits generated by Foong et al. (2019).[1,2] Data sets are split into training and test sets by uniform subsampling for the standard splits and by a custom split assigning "outer regions" to the training sets and "middle regions" to the test sets for the gap splits. For the standard splits, 20 randomized train-test splits (90% train, 10% test) of each data set are provided, with the exception of the larger *Protein* (5 splits) and *Year* (1 split) data sets. Note that the *Year* standard split is not provided in the standard splits repositories, so we assume it follows the single split (90% train, 10% test) guideline provided for that data set in the UCI respository.[3] For the gap splits, d_X train-test splits of each data set are provided, each split corresponding to one of the d_X dimensions of that data set. These splits are generated by creating "gaps" in the training data, by first sorting the data set in increasing order in the dimension of interest, then assigning the outer two-thirds to the training set and the middle third to the test set (Foong et al., 2019). The *Year* data set is not included in the gap splits repository or experiments. For each multiple split experiment, the evaluation metric is test RMSE averaged over all splits with standard error.

The total number of data points N prior to splitting and the dimensions d_X and d_Y of each provided data set are listed in Tables 3A and 3B. Note that although two of the original

1. `https://github.com/yaringal/DropoutUncertaintyExps/tree/master/UCI_Datasets`
2. `https://github.com/cambridge-mlg/DUN/tree/master/experiments/data/UCI_for_sharing`
3. `https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd`

	Hidden Layer									
Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
DNN-1	64									
DNN-2	128	64								
DNN-3	256	128	64							
DNN-5	256	128	64	32	16					
DNN-10	256	128	64	32	16	16	8	8	4	4

Table 1: Hidden layer sizes of densely-connected neural networks.

data sets—*Energy* and *Naval*—have response dimension $d_Y = 2$, all provided standard and gap splits have $d_Y = 1$. In the *Year* data set ($N = 515345$, $d_X = 90$) experiment, to make it computationally tractable, we set $d_1 = 10$ to reduce dimensionality and use the control points method with a subset ($n_S = 1000$) of training data selected as the initial n_S cluster seeds for k-means clustering according to the k-means++ algorithm.

For performance comparison with our model, we test standard ridge regression (A), NODEs, and five DNNs on the same UCI standard splits and gap splits. Ridge regression is implemented in Python with regularization weight $\lambda = 1$. We use a NODE model available on github[4], with an option for augmented or dummy dimensions (ANODE). Parameters for the NODE experiments follow those used in Dupont et al. (2019), including hidden layer size of 32, learning rate of 0.001, batch size of 256, and augmented dimension of 5 for ANODE. However, we extend the number of training epochs—or number of complete passes through the training data—to 400 to allow for model convergence which significantly improves results. The DNN models, implemented in TensorFlow and denoted DNN-x, x $= 1, 2, 3, 5, 10$, consist of x sequential densely-connected hidden layers with ReLU activation and layer sizes listed in Table 1, followed by a densely-connected output layer. In TensorFlow, we use the Adam optimizer (Kingma and Ba, 2014), MSE loss, and 400 training epochs. Default values are assumed for all other TensorFlow parameters, including learning rate of 0.001, batch size of 32, and no validation split of the data. The A, NODE, and DNN models are trained and tested on each data set split following standardization, and the standardization is removed from the model outputs for performance analysis. The NODE and DNN experiments are executed on an NVIDIA GeForce GTX 1050 GPU with Cuda 10.1. Due to size, the *Year* data set is not analyzed in the NODE experiments.

Performance is further compared with published experimental results from similarly tested state-of-the-art models found in the literature references in Table 2. The literature models include Bayesian deep learning techniques such as variational inference (VI); back-propagation (BP) and probabilistic BP (PBP) for Bayesian NNs (BNNs); Monte Carlo dropout run in a timed setting (Dropout-TS or Dropout), to convergence (Dropout-C), and with grid hyperparameter tuning (Dropout-G); BNNs with variational matrix Gaussian posteriors (VMG) and horseshoe priors (HS-BNN); and PBP with the matrix variate Gaussian distribution (PBP-MV). Additional models are Bayes by backprop (BBB); stochastic, low-rank, approximate natural-gradient (SLANG) method; variations of the neural linear (NL) model: maximum a posteriori (MAP) estimation NL (MAP NL), regularized NL (Reg

4. `https://github.com/EmilienDupont/augmented-neural-odes`

Models	**Splits**	**Reference**
VI, BP, BP-2, BP-3, BP-4, PBP, PBP-2, PBP-3, PBP-4	S	Hernández-Lobato and Adams (2015)
Dropout-TS	S	Gal and Ghahramani (2016)
VMG	D	Louizos and Welling (2016)
HS-BNN	D	Ghosh et al. (2019)
PBP-MV	D	Sun et al. (2017)
Dropout-C, Dropout-G	S	Mukhoti et al. (2018)
BBB, SLANG	S	Mishkin et al. (2018)
MAP-1, MAP-2, MAP-1 NL, MAP-2 NL, Reg-1 NL, Reg-2 NL, BN(ML)-1 NL, BN(ML)-2 NL, BN(BO)-1 NL, BN(BO)-2 NL	D,G	Ober and Rasmussen (2019)
DUN, DUN (MLP), Dropout, Ensemble, MFVI, SGD	S,G	Antoran et al. (2020)
$\mathcal{L}_{\beta-\mathrm{NLL}}$, $\beta = 0.0, 0.25, 0.5, 0.75, 1.0$, $\mathcal{L}_{\mathrm{MM}}$, $\mathcal{L}_{\mathrm{MSE}}$, Student-t, xVAMP, xVAMP*, VBEM, VBEM*	S,D	Seitzer et al. (2022)

Table 2: Literature models tested on standard splits (S), gap splits (G), and different standard splits (D).

NL), Bayesian noise (BN) NL by marginal likelihood maximization (BN(ML) NL) and by Bayesian optimization (BO) (BN(BO) NL); depth uncertainty network (DUN) with multilayer perceptron (MLP) architecture (DUN (MLP)); deep ensembles (Ensemble); Gaussian mean field VI (MFVI); vanilla NNs (SGD); and distributional regression by negative log-likelihood (NLL) with alternative loss formulation ($\beta-$NLL) ($\mathcal{L}_{\beta-\mathrm{NLL}}$), "moment matching" (MM) ($\mathcal{L}_{\mathrm{MM}}$), MSE loss ($\mathcal{L}_{\mathrm{MSE}}$), Student's t-distribution (Student-t), and different variance priors and variational inference (xVAMP, xVAMP*, VBEM, VBEM*). An integer "-x" appended to a model name denotes x hidden layers in the network.

A comprehensive list of average test RMSE results for all models in Table 2 is found in Appendix Tables C.2A and C.2B for standard split experiments and Tables C.1A and C.1B for gap split experiments. The "D" standard splits notation in Table 2 and gray shading in Tables C.2A and C.2B indicate experiments using standard splits that are different from those used in our experiments but generated following the training-test protocol from Hernández-Lobato and Adams (2015). Louizos and Welling (2016) and Sun et al. (2017) generate their own standard splits, following the training-test protocol from Hernández-Lobato and Adams (2015), and randomly generate the *Year* data split. Seitzer et al. (2022) also generate their own standard splits for the *Energy* and *Naval* data sets (maintaining the original response dimensions of $d_Y = 2$) and use the standard splits from Hernández-Lobato and Adams (2015) for the rest of the data sets. In Ghosh et al. (2019) and Ober and Rasmussen (2019), it is unclear if the standard splits are those used in Hernández-Lobato and Adams (2015) or if they are generated by the authors following that training-test protocol, thus the standard splits experiments for these models are also labelled "D" in Table 2 and their corresponding results shaded in gray in Tables C.2A and C.2B. All presented literature results involve some form of hyperparameter tuning, typically by BO or a grid approach, using a portion of each training set as a validation set. For consistency in performance comparison, we convert the standard deviation results in Ghosh et al. (2019), Antoran et al. (2020), and Seitzer et al. (2022) to standard errors and use the standard error representation of the results in Gal and Ghahramani (2016) found in Mukhoti et al. (2018). Due to size, the larger *Protein* and *Year* data sets are not analyzed in some of the literature references. All literature results are provided in 2-digit decimal precision, with the exception of 3-digit decimal precision in Hernández-Lobato and Adams (2015), Antoran et al. (2020), the *Year* analysis in Gal and Ghahramani (2016) and Louizos and Welling (2016), and the *Kin8nm* and *Wine* analysis in Seitzer et al. (2022), and 4-digit decimal precision for the *Naval* analysis in Seitzer et al. (2022).

Performance of our ADA model is compared in Tables 3A and 3B with the A, NODE, and DNN models and with a representative cross-section of the literature results in Tables C.1A, C.1B, C.2A, and C.2B from each reference in Table 2 using the same standard splits and gap splits. While all results in Tables 3A, 3B, C.1A, C.1B, C.2A, and C.2B are listed in 2-digit decimal precision, performance comparisons are carried out in higher decimal precision when necessary, if available. The lowest average test RMSE in each standard splits column and each gap splits column in Tables 3A and 3B is bolded, and result values presented in 2-digit decimal representation in the literature that cannot be confirmed as lower or higher than these lowest values are marked with a dagger symbol (†). Examples of final reshaped sequences through module D_1 of standard training splits of *Kin8nm*, *Concrete*, and *Energy* are illustrated in Figure 5. In each figure plot, data point locations represent the first three principal components of $\boldsymbol{z}^1(t)$ at a fixed time t, and color coding represents the true responses. The figure contains four plots per data set, corresponding to $t = 0.0$, 0.4, 0.7, and 1.0, respectively. Average run times for ADA, A, NODEs, and DNNs for each experiment are provided in Table 4.

	UCI Data Sets				
	Concrete	**Energy**	**Kin8nm**	**Naval**	**Power**
	$N = 1030$	$N = 768$	$N = 8192$	$N = 11934$	$N = 9568$
	$d_X = 8$	$d_X = 8$	$d_X = 8$	$d_X = 16$	$d_X = 4$
Model	$d_Y = 1$	$d_Y = 1$	$d_Y = 1$	$d_Y = 1$	$d_Y = 1$
			Standard Splits		
ADA	4.86 ± 0.12	0.50 ± 0.01	$\mathbf{0.07 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{3.39 \pm 0.05}$
A	10.31 ± 0.14	3.06 ± 0.05	0.20 ± 0.00	0.01 ± 0.00	4.61 ± 0.03
NODE	5.21 ± 0.15	0.57 ± 0.02	0.07 ± 0.00	0.00 ± 0.00	3.96 ± 0.04
ANODE	5.25 ± 0.18	0.54 ± 0.02	0.07 ± 0.00	0.00 ± 0.00	3.85 ± 0.04
DNN-1	5.02 ± 0.14	0.53 ± 0.01	0.08 ± 0.00	0.00 ± 0.00	3.98 ± 0.04
DNN-2	4.47 ± 0.13	0.51 ± 0.01	0.08 ± 0.00	0.00 ± 0.00	3.70 ± 0.04
DNN-3	$\mathbf{4.46 \pm 0.12}$	$\mathbf{0.43 \pm 0.02}$	0.08 ± 0.00	0.00 ± 0.00	3.63 ± 0.05
DNN-5	4.71 ± 0.15	0.46 ± 0.02	0.08 ± 0.00	0.00 ± 0.00	3.67 ± 0.05
DNN-10	4.64 ± 0.14	0.54 ± 0.08	0.08 ± 0.00	0.01 ± 0.00	3.59 ± 0.04
BP-3	5.57 ± 0.13	0.63 ± 0.03	0.07 ± 0.00	0.00 ± 0.00	4.11 ± 0.04
BP-4	5.53 ± 0.14	0.67 ± 0.03	0.07 ± 0.00	0.00 ± 0.00	4.18 ± 0.06
PBP-2	5.24 ± 0.12	0.90 ± 0.05	0.07 ± 0.00	0.00 ± 0.00	4.03 ± 0.03
PBP-3	5.73 ± 0.11	1.24 ± 0.06	0.07 ± 0.00	0.01 ± 0.00	4.07 ± 0.04
Dropout-TS	5.23 ± 0.12	1.66 ± 0.04	0.10 ± 0.00	0.01 ± 0.00	4.02 ± 0.04
Dropout-C	4.93 ± 0.14	1.08 ± 0.03	0.09 ± 0.00	$0.00 \pm 0.00^{\dagger}$	4.00 ± 0.04
Dropout-G	4.82 ± 0.16	0.54 ± 0.06	0.08 ± 0.00	$0.00 \pm 0.00^{\dagger}$	4.01 ± 0.04
BBB	6.16 ± 0.13	0.97 ± 0.09	0.08 ± 0.00	$0.00 \pm 0.00^{\dagger}$	4.21 ± 0.03
SLANG	5.58 ± 0.19	0.64 ± 0.03	0.08 ± 0.00	$0.00 \pm 0.00^{\dagger}$	4.16 ± 0.04
DUN (MLP)	4.57 ± 0.16	0.95 ± 0.11	0.08 ± 0.00	0.00 ± 0.00	3.67 ± 0.06
Dropout	4.61 ± 0.13	0.57 ± 0.05	0.07 ± 0.00	0.00 ± 0.00	3.82 ± 0.08
Ensemble	4.55 ± 0.13	0.51 ± 0.02	0.30 ± 0.22	0.00 ± 0.00	3.44 ± 0.05
$\mathcal{L}_{\mathbf{MSE}}$	4.96 ± 0.14	$--$	0.08 ± 0.00	$--$	4.01 ± 0.04
VBEM*	5.17 ± 0.13	$--$	0.08 ± 0.00	$--$	4.02 ± 0.04
			Gap Splits		
ADA	7.61 ± 0.38	3.51 ± 1.20	$\mathbf{0.07 \pm 0.00}$	0.02 ± 0.00	5.33 ± 0.43
A	10.75 ± 0.29	3.96 ± 0.36	0.20 ± 0.00	0.03 ± 0.00	4.47 ± 0.08
NODE	7.77 ± 0.27	4.98 ± 1.82	0.08 ± 0.00	0.03 ± 0.00	4.39 ± 0.11
ANODE	8.49 ± 0.30	4.90 ± 1.99	0.07 ± 0.00	0.02 ± 0.00	4.36 ± 0.15
DNN-1	7.53 ± 0.34	4.59 ± 1.75	0.08 ± 0.00	0.03 ± 0.00	4.33 ± 0.13
DNN-2	7.45 ± 0.31	3.77 ± 1.34	0.08 ± 0.00	0.03 ± 0.00	5.17 ± 0.33
DNN-3	7.44 ± 0.23	3.93 ± 1.42	0.08 ± 0.00	0.03 ± 0.00	5.67 ± 0.33
DNN-5	7.28 ± 0.16	3.23 ± 1.08	0.08 ± 0.00	0.03 ± 0.00	5.78 ± 0.41
DNN-10	8.41 ± 1.05	5.98 ± 1.42	0.08 ± 0.00	0.02 ± 0.00	5.56 ± 0.45
MAP-1	7.79 ± 0.18	$\mathbf{2.83 \pm 0.99}$	0.09 ± 0.01	0.02 ± 0.00	$\mathbf{4.24 \pm 0.12}$
MAP-2	7.78 ± 0.23	3.70 ± 1.33	0.08 ± 0.00	0.03 ± 0.00	4.33 ± 0.18
MAP-2 NL	7.44 ± 0.17	3.48 ± 1.21	$0.07 \pm 0.00^{\dagger}$	0.03 ± 0.00	4.27 ± 0.08
BN(ML)-2 NL	7.33 ± 0.36	4.10 ± 1.64	0.08 ± 0.00	$\mathbf{0.01 \pm 0.00}$	5.17 ± 0.28
DUN	7.20 ± 0.18	2.94 ± 0.67	0.08 ± 0.00	0.02 ± 0.00	4.30 ± 0.09
Dropout	7.06 ± 0.21	2.87 ± 0.50	0.07 ± 0.00	0.03 ± 0.00	4.69 ± 0.07
Ensemble	$\mathbf{6.85 \pm 0.18}$	3.36 ± 0.83	1.63 ± 0.99	0.02 ± 0.00	4.37 ± 0.09
MFVI	7.55 ± 0.19	8.61 ± 2.10	0.10 ± 0.01	0.03 ± 0.01	4.68 ± 0.16

† Literature result indistinguishable from the best value.

Table 3A: Average test RMSE $\pm$ 1 standard error (best values in bold) for ADA, A, NODEs, DNNs, and a cross-section of the literature models in Tables C.1A and C.2A.

	UCI Data Sets			
	Protein	**Wine**	**Yacht**	**Year**
	$N = 45730$	$N = 1599$	$N = 308$	$N = 515345$
	$d_X = 9$	$d_X = 11$	$d_X = 6$	$d_X = 90$
Model	$d_Y = 1$	$d_Y = 1$	$d_Y = 1$	$d_Y = 1$
	Standard Splits			
ADA	$\mathbf{3.24 \pm 0.01}$	$\mathbf{0.59 \pm 0.01}$	0.72 ± 0.06	$8.88 \pm \mathrm{NA}$
A	5.21 ± 0.02	0.65 ± 0.01	8.95 ± 0.27	$9.51 \pm \mathrm{NA}$
NODE	4.11 ± 0.02	0.71 ± 0.01	1.16 ± 0.09	$--$
ANODE	4.08 ± 0.02	0.71 ± 0.01	0.93 ± 0.10	$--$
DNN-1	4.35 ± 0.04	0.66 ± 0.01	0.95 ± 0.07	$8.96 \pm \mathrm{NA}$
DNN-2	3.79 ± 0.02	0.73 ± 0.02	0.92 ± 0.08	$9.78 \pm \mathrm{NA}$
DNN-3	3.83 ± 0.03	0.66 ± 0.02	1.27 ± 0.15	$10.73 \pm \mathrm{NA}$
DNN-5	3.70 ± 0.01	0.64 ± 0.01	1.22 ± 0.12	$10.20 \pm \mathrm{NA}$
DNN-10	4.73 ± 0.52	0.70 ± 0.02	3.32 ± 1.02	$10.31 \pm \mathrm{NA}$
BP-3	4.01 ± 0.03	0.65 ± 0.01	1.11 ± 0.09	$8.93 \pm \mathrm{NA}$
BP-4	3.96 ± 0.01	0.65 ± 0.02	1.27 ± 0.13	$9.05 \pm \mathrm{NA}$
PBP-2	4.25 ± 0.02	0.64 ± 0.01	0.85 ± 0.05	$8.92 \pm \mathrm{NA}$
PBP-3	4.09 ± 0.03	0.64 ± 0.01	0.89 ± 0.10	$8.87 \pm \mathrm{NA}$
Dropout-TS	4.36 ± 0.01	0.62 ± 0.01	1.11 ± 0.09	$\mathbf{8.85} \pm \mathrm{NA}$
Dropout-C	4.27 ± 0.01	0.61 ± 0.01	0.70 ± 0.05	$--$
Dropout-G	4.27 ± 0.02	0.62 ± 0.01	0.67 ± 0.05	$--$
BBB	$--$	0.64 ± 0.01	1.13 ± 0.06	$--$
SLANG	$--$	0.65 ± 0.01	1.08 ± 0.06	$--$
DUN (MLP)	3.41 ± 0.03	0.63 ± 0.01	2.47 ± 0.19	$--$
Dropout	3.43 ± 0.03	0.64 ± 0.01	0.88 ± 0.09	$--$
Ensemble	3.26 ± 0.03	1.93 ± 1.28	1.43 ± 0.11	$--$
$\mathcal{L}_{\mathbf{MSE}}$	4.28 ± 0.03	0.63 ± 0.01	0.78 ± 0.06	$--$
VBEM*	4.35 ± 0.04	0.63 ± 0.01	$\mathbf{0.65 \pm 0.04}$	$--$
	Gap Splits			
ADA	5.13 ± 0.20	0.68 ± 0.01	1.02 ± 0.14	
A	5.34 ± 0.04	0.64 ± 0.01	9.24 ± 0.31	
NODE	5.46 ± 0.15	0.79 ± 0.01	2.71 ± 0.29	
ANODE	5.49 ± 0.20	0.84 ± 0.01	2.29 ± 0.52	
DNN-1	5.08 ± 0.09	0.72 ± 0.01	2.33 ± 0.29	
DNN-2	5.56 ± 0.20	0.81 ± 0.01	3.40 ± 0.64	
DNN-3	5.95 ± 0.21	0.73 ± 0.01	3.53 ± 0.59	
DNN-5	5.85 ± 0.24	0.74 ± 0.01	3.29 ± 0.54	
DNN-10	6.04 ± 0.21	0.76 ± 0.01	3.95 ± 0.71	
MAP-1	5.16 ± 0.04	$0.63 \pm 0.01^{\dagger}$	1.31 ± 0.14	
MAP-2	5.07 ± 0.06	$0.63 \pm 0.01^{\dagger}$	1.05 ± 0.09	
MAP-2 NL	5.08 ± 0.06	$0.63 \pm 0.01^{\dagger}$	$\mathbf{1.01 \pm 0.09}$	
BN(ML)-2 NL	5.37 ± 0.17	0.64 ± 0.01	1.31 ± 0.16	
DUN	5.21 ± 0.35	0.70 ± 0.01	1.85 ± 0.17	
Dropout	5.13 ± 0.28	0.66 ± 0.01	2.29 ± 0.47	
Ensemble	$\mathbf{4.80 \pm 0.27}$	0.67 ± 0.01	1.84 ± 0.19	
MFVI	5.12 ± 0.13	$\mathbf{0.63 \pm 0.01}$	1.84 ± 0.16	

† Literature result indistinguishable from the best value.

Table 3B: Average test RMSE $\pm$ 1 standard error (best values in bold) for ADA, A, NODEs, DNNs, and a cross-section of the literature models in Tables C.1B and C.2B.

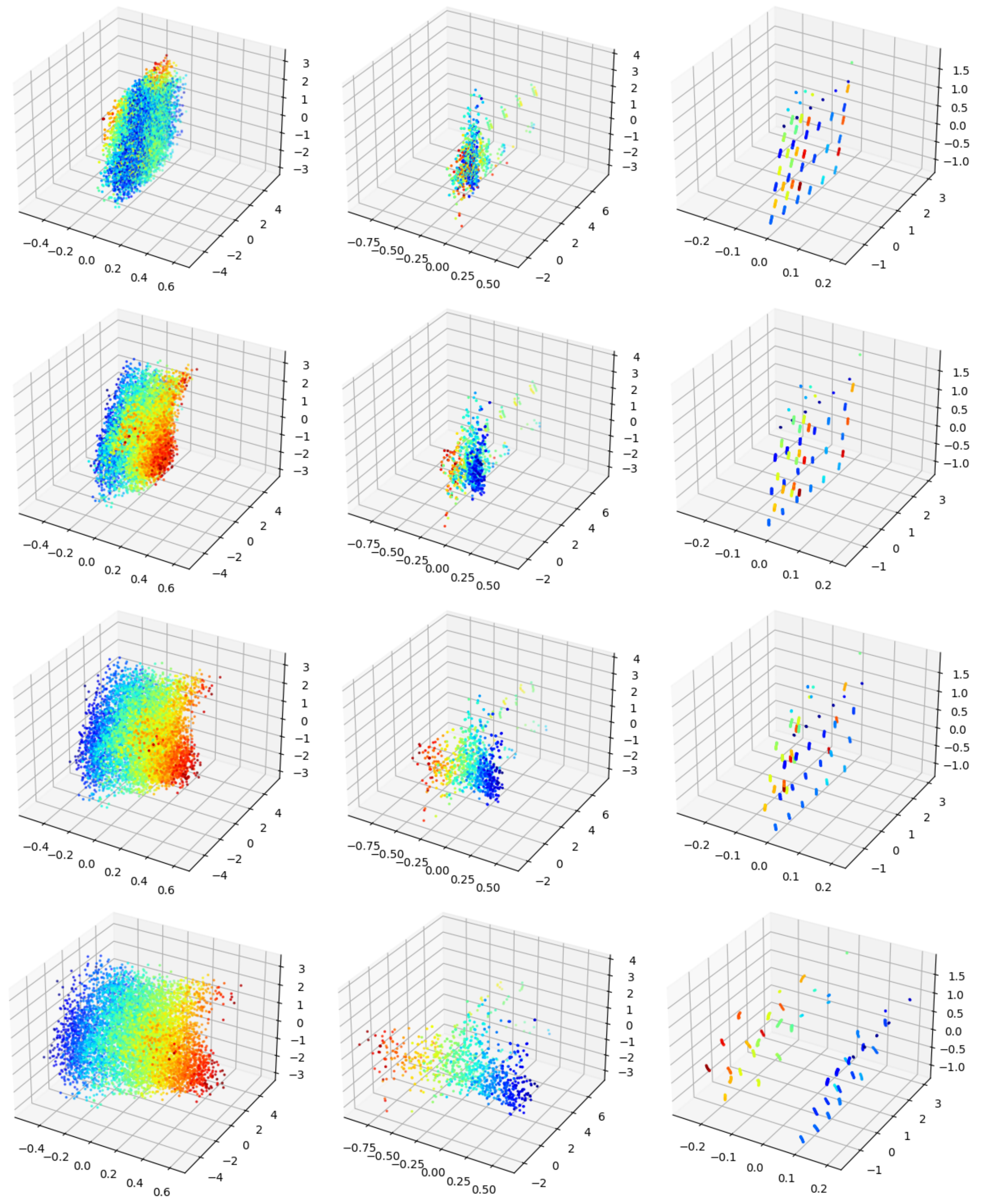

Figure 5: Reshaped sequences of *Kin8nm* (left), *Concrete* (middle), and *Energy* (right) standard training splits through module D_1. Data points correspond to the first three principal components of $\boldsymbol{z}^1(t)$ at a fixed time t, with colors representing the true responses. Starting from the top, $t = 0.0$, 0.4, 0.7, and 1.0, respectively.

	UCI Data Sets								
Model	**Concrete**	**Energy**	**Kin8nm**	**Naval**	**Power**	**Protein**	**Wine**	**Yacht**	**Year**
				Standard Splits					
ADA	3:24	1:26	12:21	28:41	27:19	3:57:53	4:27	0:58	22:16:39
A	0:00	0:00	0:00	0:00	0:00	0:00	0:00	0:00	0:01
NODE	0:34	0:25	5:03	6:19	5:33	29:13	0:53	0:17	−−
ANODE	0:34	0:25	4:24	6:38	4:50	28:22	0:51	0:17	−−
DNN-1	0:09	0:07	1:03	1:33	1:16	6:15	0:13	0:03	1:07:04
DNN-2	0:09	0:07	1:15	1:42	1:24	6:28	0:14	0:03	1:14:57
DNN-3	0:10	0:08	1:20	1:51	1:35	6:59	0:15	0:04	1:19:55
DNN-5	0:11	0:09	1:29	2:12	1:47	7:52	0:17	0:04	1:31:06
DNN-10	0:15	0:11	1:55	2:46	2:18	10:15	0:22	0:05	1:52:08
				Gap Splits					
ADA	2:05	1:30	9:33	21:27	21:35	2:32:31	3:44	0:55	
A	0:00	0:00	0:00	0:00	0:00	0:00	0:00	0:00	
NODE	0:25	0:17	3:28	4:43	3:49	21:06	0:43	0:09	
ANODE	0:25	0:17	3:08	4:42	3:32	21:27	0:42	0:08	
DNN-1	0:07	0:04	0:48	1:09	0:55	4:19	0:11	0:03	
DNN-2	0:07	0:04	0:52	1:15	1:01	4:47	0:12	0:03	
DNN-3	0:08	0:05	0:56	1:21	1:06	5:09	0:13	0:03	
DNN-5	0:09	0:06	1:04	1:33	1:15	6:09	0:15	0:04	
DNN-10	0:12	0:07	1:22	2:00	1:36	8:03	0:18	0:05	

Table 4: Average run times for the baseline experiments. Times are in mm:ss or hh:mm:ss format (h: hour; m: minute; s: second), as applicable.

15. Discussion

In comparison with all models tested on the same UCI standard splits in Tables 3A, 3B, C.2A, and C.2B, our baseline ADA model performs well in general, with best performance ranking or lowest average test RMSE for five data sets—*Kin8nm*, *Naval*, *Power*, *Protein*, and *Wine*—third lowest for *Energy*, and fourth lowest for *Yacht* and *Year*. We note that for the *Naval* standard splits, it is unclear if the literature results for models Dropout-C, Dropout-G, BBB, and SLANG are better or worse than the otherwise top-performing result of our model for that experiment. Additionally, the use of the control points method and dimensionality reduction for the *Year* experiment enabled experiment tractability with competitive results. The DNN-3 model outperforms all models for the *Concrete* and *Energy* data sets, and VBEM* and Dropout-TS have the lowest average RMSEs for *Yacht* and *Year*, respectively. In comparison with all models tested on different standard splits in Tables C.2A and C.2B and with results in high enough decimal precision for comparison, our ADA model maintains a similar performance level, as those literature model experiments primarily show improved performance for the *Energy*, *Yacht*, and *Year* data sets on which ADA did not have top performance. Sample D_1 deformation sequences in Figure 5 for several standard split experiments demonstrate the smooth invertibility of data transformations through the D modules of our model.

While the standard splits are useful for testing a model's ability to fit data, the gap splits can test in a sense how well a model generalizes to out-of-distribution data. A robust model will simultaneously perform well on the standard splits and not critically fail on the gap splits. Our ADA model demonstrates above average performance overall in ranking comparisons with all models tested on the UCI gap splits in Tables 3A, 3B, C.1A, and C.1B, with no excessively poor predictions aside from one low performance ranking (22nd out of 25 models) for *Power*. Specifically, in comparison with the other models, our model's average test RMSE is the lowest for *Kin8nm*, second lowest for *Yacht*, sixth lowest for *Protein*, and ninth lowest for *Energy* and *Naval*, with approximately center performance ranking for *Concrete* and *Wine*. Our model's top ranking result for the *Kin8nm* gap splits is caveated with the fact that the literature results for models MAP-2 NL, Reg-2 NL, and BN(BO)-2 NL cannot be confirmed as lower or higher than this value. In addition, the ADA results cannot be distinguished from the results for MAP-1, MAP-1 NL, and DUN (MLP) for *Naval* gap splits and from the results for MAP-1 NL for *Protein* gap splits.

Average run time comparisons of ADA, A, NODE, and DNN models in Table 4 demonstrate the computational trade-off for the higher performance of ADA. Among these baseline models, the fastest run times in all experiments is achieved by the A model, followed by DNNs, then NODEs, then the ADA model.

A summary of ADA performance rankings and average run times for both the standard split and gap split experiments are found in Table 7. Rankings are in comparison with all models in Tables 3A, 3B, C.1A, C.1B, C.2A, and C.2B using the same standard splits and gap splits. Run times are in minutes, rounded to the nearest integer.

16. Beyond the Baseline

We test more complex model architectures beyond the ADA baseline in Table 5, using the *Airfoil* data set in the UCI repository for the experiments. Performance is compared with our baseline model as well as DNNs, as the *Airfoil* data set has similar size and dimensions to the *Concrete* and *Energy* data sets on which the DNN models perform best in Table 3A. We generate 10 randomized train-test splits (90% train, 10% test) and again evaluate prediction performance on the test splits. The hyperparameter values of the diffeomorphic regression models in Table 5 follow those used in the experiments in Section 14, including the same A and D modules used in our baseline model, with one exception. For AD$^{\text{x}}$A models with x sequential D modules, while the D modules have the same dimension, kernel type, and number of discretized time points as in our ADA model, the kernel widths of this sequence of m D modules increase as $h_q = \frac{q}{m+1}$, $q = 1, \ldots, m$, or decrease as $h_q = \frac{m-q+1}{m+1}$, $q = 1, \ldots, m$. All other experimental settings in Table 5 follow Section 14. Starting with the original ADA model, the results show improved performance with increased model complexity, with the AD^{4}A model with decreasing kernel sizes outperforming all models, including the DNNs, which again outperform the baseline. Note that to ensure these improved results are not the result of the increased number of time steps inherent in increasing the number of our D modules, the ADA model is also run with increased T_1, as shown in Table 5.

We test the AD^{4}A model further in Table 6 on the standard and gap splits of six of the UCI data sets, using the same experimental settings in Section 14, and summarize its performance rankings and average run times in Table 7. In comparison with ADA in

Model	**UCI Data Set** **Airfoil** $N = 1503$ $d_X = 5$ $d_Y = 1$
ADA	1.49 ± 0.05
ADA ($T_1 = 20$)	1.49 ± 0.05
ADA ($T_1 = 30$)	1.45 ± 0.04
ADA ($T_1 = 40$)	1.40 ± 0.08
DNN-1	2.02 ± 0.06
DNN-2	1.36 ± 0.05
DNN-3	1.25 ± 0.05
DNN-5	$\mathbf{1.16 \pm 0.04}$
DNN-10	1.79 ± 0.60

Model	**UCI Data Set** **Airfoil** $N = 1503$ $d_X = 5$ $d_Y = 1$
AD^{2}A ($h_q \uparrow$)	1.34 ± 0.06
AD^{3}A ($h_q \uparrow$)	1.20 ± 0.04
AD^{4}A ($h_q \uparrow$)	1.11 ± 0.04
AD^{2}A ($h_q \downarrow$)	1.18 ± 0.04
AD^{3}A ($h_q \downarrow$)	1.07 ± 0.03
AD^{4}A ($h_q \downarrow$)	$\mathbf{1.02 \pm 0.03}$
(AD)^{2}A	1.38 ± 0.07
(AD)^{3}A	1.29 ± 0.06
(AD)^{4}A	1.26 ± 0.05
(AD)^{5}A	1.33 ± 0.05

Table 5: Average test RMSE ± 1 standard error (best values in bold). Increasing and decreasing kernel sizes are denoted ($h_q \uparrow$) and ($h_q \downarrow$), respectively.

Model	**UCI Data Sets** **Concrete**	**Energy**	**Kin8nm**	**Power**	**Wine**	**Yacht**
			Standard Splits			
ADA	4.86 ± 0.12	$\mathbf{0.50 \pm 0.01}$	0.07 ± 0.00	3.39 ± 0.05	0.59 ± 0.01	$\mathbf{0.72 \pm 0.06}$
AD^{4}A	$\mathbf{4.49 \pm 0.13}$	0.51 ± 0.01	$\mathbf{0.07 \pm 0.00}$	$\mathbf{3.27 \pm 0.06}$	$\mathbf{0.58 \pm 0.01}$	0.78 ± 0.05
			Gap Splits			
ADA	7.61 ± 0.38	$\mathbf{3.51 \pm 1.20}$	0.07 ± 0.00	5.33 ± 0.43	0.68 ± 0.01	$\mathbf{1.02 \pm 0.14}$
AD^{4}A	$\mathbf{7.29 \pm 0.37}$	3.52 ± 1.19	$\mathbf{0.07 \pm 0.00}$	$\mathbf{4.67 \pm 0.21}$	$\mathbf{0.68 \pm 0.01}$	1.04 ± 0.18

Table 6: Average test RMSE ± 1 standard error (best values in bold) for ADA and AD^{4}A with decreasing kernel sizes.

Table 6, the more complex model shows improved (*Concrete*, *Kin8nm*, *Power*, *Wine*) and similar (*Energy*) test RMSE results for both sets of experiments, with slightly worse results on the remaining *Yacht* experiments. In particular, for the standard splits, the improved *Concrete* result in Table 6 is now comparable with the best (lowest) result from DNN-3. In comparison with the literature, we note the AD^{4}A results cannot be distinguished from the results for $\mathcal{L}_{\text{MSE}}$ for *Yacht* standard splits, and AD^{4}A and ADA share the same results caveat for *Kin8nm* gap splits. In comparison with ADA in Table 7, AD^{4}A has similar performance rankings, with the exception of significant improvements for *Concrete* (from ten to three for standard splits and from 13 to five for gap splits) and for *Power* gap splits (from 22 to 15 or low to approximately center ranking), thus demonstrating even greater overall robustness.

	Ranking		Avg. Run Time (minutes)		Ranking		Avg. Run Time (minutes)	
Data Set	**ADA**	**AD^{4}A**	**ADA**	**AD^{4}A**	**ADA**	**AD^{4}A**	**ADA**	**AD^{4}A**
	Standard Splits				**Gap Splits**			
Concrete	10/41	3/41	3	10	13/25	5/25	2	6
Energy	3/29	4/29	1	5	9/25	9/25	2	4
Kin8nm	1/41	1/41	12	41	1-4/25†	1-4/25†	10	30
Naval	1-5/29†	−−	29	−−	9-12/25†	−−	21	−−
Power	1/41	1/41	27	56	22/25	15/25	22	41
Protein	1/39	−−	238	−−	6-7/25†	−−	153	−−
Wine	1/41	1/41	4	12	15/25	15/25	4	13
Yacht	4/41	4-5/41†	1	3	2/25	2/25	1	4
Year	4/17	−−	1337	−−				

† Literature results indistinguishable from the ADA or AD^{4}A result included in the ranking range.

Table 7: Test performance rankings and average run times of ADA and AD^{4}A with decreasing kernel sizes. Rankings are in comparison with all A, NODEs, and DNNs in Tables 3A and 3B and literature results in Tables C.1A, C.1B, C.2A, and C.2B using the same standard splits and gap splits. Times are in minutes, rounded to the nearest integer.

17. Summary

We present a layered approach to multivariate regression using FineMorphs, a sequence model of affine and diffeomorphic transformations. Optimal control concepts from shape analysis are leveraged to optimally "reshape" model states while learning. Diffeomorphisms (the model states) are generated by RKHS time-dependent vector fields (the control) calculated by Hamiltonian control theory, minimizing—along with the optimal affine parameters—a learning objective functional consisting of a kinetic energy term and affine and endpoint costs. In our setting, any arbitrary number and order of arbitrary affine and diffeomorphic transformations can be modeled, and diffeomorphisms can be generated as flows of sub-optimal vector fields parametrized by control points to reduce computational complexity and enable training on large data sets. Additionally, the affine modules can scale data prior to diffeomorphic transforms as well as reduce (or increase) dimensionality. For both the optimal and sub-optimal vector fields cases, a proof of the existence of a solution to the variational problem and a derivation of the necessary conditions for optimality are provided. On standard UCI benchmark experiments, our baseline diffeomorphic regression model—ADA—performs favorably overall against state-of-the-art, hyperparameter-tuned deep BNNs and other models in the literature as well as NODEs and TensorFlow DNNs. The computational intractability of the largest data set in the experiments is successfully addressed with our model's dimensionality reduction and control points capabilities, with good performance. A general trend of improved performance with increased model complexity is observed, in particular with "coarse-to-fine" models containing multiple sequential diffeomorphisms of decreasing kernel sizes. Additionally, our models demonstrate out-of-distribution robustness with reasonable performances in experiments using custom train-test splits with “gaps” in the training data. In contrast with other methods, our diffeomorphic regression models learn smooth, invertible transformations of the shapes or manifolds on which the data sets—and deeper model states—lie, with explicit control of the smoothness of these transformations. Future work includes further understanding of the theoretical basis, limitations, and advantages of our models; investigating coarse-to-fine architectures and Riemannian optimization methods on the diffeomorphism group (Boumal, 2023); and extending the diffeomorphic learning model to manifold learning.

Appendix A. Existence of Solution to the FineMorphs Variational Problem

The variational problem in (3) is to minimize

$$G(v_1,\ldots,v_m,M_0,\ldots,M_m,b_0,\ldots,b_m) = \sum_{q=1}^{m} \|v_q\|^2_{\mathcal{H}_q} + \lambda \sum_{q=0}^{m} \|M_q\|^2 + \frac{1}{\sigma^2}\sum_{k=1}^{n} \Gamma_k(\pi_r(\zeta_k^{m+1})) \tag{5}$$

over $v_q \in \mathcal{H}_q$, $q = 1,\ldots,m$, and $M_q \in \mathcal{M}_{d_{q+1},d_q}(\mathbb{R})$, $b_q \in \mathbb{R}^{d_{q+1}}$, $q = 0,\ldots,m$, subject to

$$\begin{cases} \partial_t \boldsymbol{\varphi}_{v_q}(t) = v_q(t) \circ \boldsymbol{\varphi}_{v_q}(t), \quad t \in [0,1] \\ \quad \zeta_k^{q+1} = M_q \boldsymbol{\varphi}_{v_q}(1, \zeta_k^q) + b_q \\ \boldsymbol{\varphi}_{v_q}(0) = \mathrm{id}, \quad \zeta_k^1 = M_0 \iota_s(x_k) + b_0. \end{cases} \tag{6}$$

G is bounded from below and thus has an infimum $G_{\min}$. We want to show that $G_{\min}$ is also a minimum, i.e., there exists $v_q^{(*)} \in \mathcal{H}_q$, $q = 1,\ldots,m$, and $M_q^{(*)} \in \mathcal{M}_{d_{q+1},d_q}(\mathbb{R})$, $b_q^{(*)} \in \mathbb{R}^{d_{q+1}}$, $q = 0,\ldots,m$, such that

$$G(v_1^{(*)},\ldots,v_m^{(*)},M_0^{(*)},\ldots,M_m^{(*)},b_0^{(*)},\ldots,b_m^{(*)}) = G_{\min}.$$

We prove this under the following weak assumptions which are satisfied in all practical cases within our problem space. We introduce the following action of translation on diffeomorphisms: $b \cdot \varphi_q : x \mapsto \varphi(x+b) - b$ and corresponding infinitesimal action on vector fields $b \cdot f : x \mapsto f(x+b)$. (As is customary, we use the same notation for action and infinitesimal action.)

(H1) The Hilbert norms on V_q (recall that $\mathcal{H}_q = L^2([0,1], V_q)$) are translation invariant: for any $f \in V_q$ and $b \in \mathbb{R}^{d_q}$, the vector field $b \cdot f$ belongs to V_q with $\|b \cdot f\|_{V_q} = \|f\|_{V_q}$.

(H2) The functions Γ_k are continuous, non-negative, and satisfy $\Gamma_k(\zeta) \to \infty$ when $\|\zeta\| \to \infty$.

The existence proof is detailed below, first in the unconstrained case of equation (5), then in the "sub-Riemannian" case introduced in Section 8.

Existence: unconstrained case. If $v \in \mathcal{H}_q$ and $b \in \mathbb{R}^{d_q}$, we will denote by $b \cdot v$ the time-dependent vector field $t \mapsto b \cdot v(t)$. Importantly, the associated flow $\boldsymbol{\varphi}_{b \cdot v}$ (defined by $\partial_t \boldsymbol{\varphi}_{b\cdot v}(t,x) = b \cdot v(t)(\boldsymbol{\varphi}_{b \cdot v}(t,x))$ and $\boldsymbol{\varphi}_{b\cdot v}(0,x) = x$) satisfies $\boldsymbol{\varphi}_{b\cdot v} = b \cdot \boldsymbol{\varphi}_v$.

Indeed $b \cdot \boldsymbol{\varphi}_v(0,x) = \boldsymbol{\varphi}_v(0, x+b) - b = x$, and

$$\begin{aligned} \partial_t(b \cdot \boldsymbol{\varphi}_v)(t,x) &= \partial_t(\boldsymbol{\varphi}_v(t,x+b) - b) \\ &= v(t)(\boldsymbol{\varphi}_v(t,x+b)) \\ &= b \cdot v(t)(\boldsymbol{\varphi}_v(t,x+b) - b) = b \cdot v(t)((b \cdot \boldsymbol{\varphi}_v)(t,x)). \end{aligned}$$

As a consequence, if (H1) is true, one can assume, without changing the value of the infimum, that $b_0 = \cdots = b_{m-1} = 0$. Indeed, given $v_1, \ldots, v_m$, $M_0, \ldots, M_m$, $b_0, \ldots, b_m$, one can define

$$c_q = \begin{cases} b_0, & \text{if } q = 0 \\ M_q c_{q-1} + b_q, & \text{if } 1 \leq q \leq m, \end{cases}$$

and, one has, letting $\tilde{v}_q = c_{q-1} \cdot v_q$, $q \geq 1$, $\tilde{b}_0 = \cdots = \tilde{b}_{m-1} = 0$, $\tilde{b}_m = c_m$,

$$G(\tilde{v}_1, \ldots, \tilde{v}_m, M_0, \ldots, M_m, 0, \cdots, 0, \tilde{b}_m) = G(v_1, \ldots, v_m, M_0, \ldots, M_m, b_0, \ldots, b_m). \tag{7}$$

To see this, let $\boldsymbol{\varphi}_{\tilde{v}_q}$ and $\tilde{\zeta}_k^q$ be defined by (6), i.e.,

$$\begin{cases} \partial_t \boldsymbol{\varphi}_{\tilde{v}_q}(t) = \tilde{v}_q(t) \circ \boldsymbol{\varphi}_{\tilde{v}_q}(t), \quad t \in [0,1] \\ \qquad \tilde{\zeta}_k^{q+1} = M_q \boldsymbol{\varphi}_{\tilde{v}_q}(1, \tilde{\zeta}_k^q) + \tilde{b}_q \\ \boldsymbol{\varphi}_{\tilde{v}_q}(0) = \mathrm{id}, \quad \tilde{\zeta}_k^1 = M_0 \iota_s(x_k) + \tilde{b}_0. \end{cases}$$

As we just saw, we have $\boldsymbol{\varphi}_{\tilde{v}_q} = c_{q-1} \cdot \boldsymbol{\varphi}_{v_q}$. Moreover, for $q \leq m-1$ (so that $\tilde{b}_q = 0$), we have

$$\tilde{\zeta}_k^{q+1} = M_q(c_{q-1} \cdot \boldsymbol{\varphi}_{v_q})(1, \tilde{\zeta}_k^q) = M_q \boldsymbol{\varphi}_{v_q}(1, \tilde{\zeta}_k^q + c_{q-1}) - M_q c_{q-1},$$

yielding

$$\tilde{\zeta}_k^{q+1} + c_q = M_q \boldsymbol{\varphi}_{v_q}(1, \tilde{\zeta}_k^q + c_{q-1}) + b_q.$$

So $\tilde{\zeta}_k^{q+1} + c_q$ satisfy the same iterations as ζ_k^{q+1}, with same initial condition

$$\tilde{\zeta}_k^1 + c_0 = M_0 \iota_s(x_k) + c_0 = \zeta_k^1$$

(since $c_0 = b_0$). This shows that $\tilde{\zeta}_k^{q+1} + c_q = \zeta_k^{q+1}$, $q = 0, \ldots, m-1$. Finally,

$$\tilde{\zeta}_k^{m+1} = M_m \boldsymbol{\varphi}_{v_m}(1, \tilde{\zeta}_k^m + c_{m-1}) - M_m c_{m-1} + c_m = M_m \boldsymbol{\varphi}_{v_m}(1, \zeta_k^m) + b_m = \zeta_k^{m+1}.$$

Since $\|\tilde{v}_q\|_{\mathcal{H}_q} = \|v_q\|_{\mathcal{H}_q}$, (7) is satisfied.

We now conclude the argument by considering a minimizing sequence for G in the form $v_q^{(\ell)} \in \mathcal{H}_q$, $q = 1, \ldots, m$, and $M_q^{(\ell)} \in \mathcal{M}_{d_{q+1}, d_q}(\mathbb{R})$, $b_q^{(\ell)} \in \mathbb{R}^{d_{q+1}}$, $q = 0, \ldots, m$, with $b_0^{(\ell)} = \cdots = b_{q-1}^{(\ell)} = 0$, satisfying

$$\lim_{\ell \to \infty} G(v_1^{(\ell)}, \ldots, v_m^{(\ell)}, M_0^{(\ell)}, \ldots, M_m^{(\ell)}, b_0^{(\ell)}, \ldots, b_m^{(\ell)}) = G_{\min}.$$

Denote by $\boldsymbol{\varphi}_{v_q^{(\ell)}}$ and $\zeta_k^{q(\ell)}$ the diffeomorphisms and vectors defined in (6) for each ℓ.

Each $M_q^{(\ell)}$ sequence is bounded in $\mathcal{M}_{d_{q+1}, d_q}(\mathbb{R})$, and each $v_q^{(\ell)}$ is bounded in $\mathcal{H}_q$. There is therefore no loss of generality (just using a subsequence) in assuming that $M_q^{(\ell)}$ converges to some $M_q^{(*)} \in \mathcal{M}_{d_{q+1}, d_q}(\mathbb{R})$, and that $v_q^{(\ell)}$ converges weakly in $\mathcal{H}_q$ to some $v_q^{(*)}$ that satisfies

$$\liminf_{\ell \to \infty} \|v_q^{(\ell)}\|_{\mathcal{H}_q}^2 \geq \|v_q^{(*)}\|_{\mathcal{H}_q}^2. \tag{8}$$

Let $\boldsymbol{\varphi}_{v_q^{(*)}}(t)$ be the flow associated with $v_q^{(*)}$. Weak convergence in $\mathcal{H}_q$ implies, at each fixed $t \in [0,1]$, uniform convergence of $\boldsymbol{\varphi}_{v_q^{(\ell)}}(t)$ to $\boldsymbol{\varphi}_{v_q^{(*)}}(t)$ on $\mathbb{R}^{d_q}$ (Younes, 2010). As a consequence, for all $q \leq m$, the sequence $(\zeta_k^{q(\ell)}, \ell \geq 0)$ also converges to a limit $\zeta_k^{q(*)}$ that satisfies (6).

Each $\Gamma_k(\pi_r(\zeta_k^{m+1(\ell)}))$ must be bounded independently of ℓ, since we have a minimizing sequence. Assumption (H2) then implies that $\pi_r(\zeta_k^{m+1(\ell)})$ is also bounded, with $\pi_r(\zeta_k^{m+1(\ell)}) = \pi_r(M_m^{(\ell)} \boldsymbol{\varphi}_{v_m^{(\ell)}}(1, \zeta_k^{m(\ell)})) + \pi_r(b_m^{(\ell)})$. Since the first term in this sum converges, we see that $\pi_r(b_m^{(\ell)})$ is also bounded, so that, taking a subsequence if needed, we can assume that $\pi_r(b_m^{(\ell)})$ converges to some $b_m^{(*)}$ (with $\pi_r(b_m^{(*)}) = b_m^{(*)}$). Using (8), we obtain

$$G(v_1^{(*)}, \ldots, v_m^{(*)}, M_0^{(*)}, \ldots, M_m^{(*)}, 0, \ldots, 0, b_m^{(*)}) = G_{\min},$$

which concludes the proof.

Existence: sub-Riemannian case. The situation in which the vector fields v_q are restricted to sub-optimal finite-dimensional spaces, as considered in Section 8, is handled similarly, and follows arguments previously made in Younes (2012); Arguillere et al. (2015); Gris et al. (2018); Younes et al. (2020). Here, we associate a closed subspace of V_q with a diffeomorphism ψ on $\mathbb{R}^{d_q}$. This subspace will also depend on the configuration (denoted ζ^q) that comes as input to the diffeomorphic module. We denote this subspace as $W_q(\psi, \zeta)$, with $\psi \in \mathrm{Diff}_{V_q}$ and $\zeta \in (\mathbb{R}^{d_q})^n$. We also denote the orthonormal projection of $f \in V_q$ onto $W_q(\psi, \zeta)$ as $P_{W_q(\psi,\zeta)}(f)$.

We will make the following hypotheses on the spaces $W_q(\psi, \zeta)$, which form a "distribution" in the terminology of sub-Riemannian geometry.

(HS1) For $b \in \mathbb{R}^{d_q}$, let $b \cdot W_q(\psi, \zeta) = \{b \cdot f : f \in W_q(\psi, \zeta)\}$. We assume $b \cdot W_q(\psi, \zeta) = W_q(b \cdot \psi, \zeta - b)$.

(HS2) The spaces $W_q(\psi, \zeta)$ depend continuously on ψ and ζ, in the sense that the mapping $\psi \mapsto P_{W_q(\psi,\zeta)}$, which takes values in the space of linear operators on V_q, is continuous in ψ (for uniform convergence) and ζ.

In the setting of equation (5), we now add to the minimization the requirement that each v_q belongs to the space

$$\mathcal{W}_q(\boldsymbol{\varphi}_{v_q}(\cdot), \zeta^q) = \left\{ v \in \mathcal{H}_q : v(t) \in W_q(\boldsymbol{\varphi}_{v_q}(t), \zeta^q) \text{ for almost all } t \in [0,1] \right\}.$$

Then, assuming (H1), (H2), (HS1) and (HS2), there exists a solution to this minimization problem.

The proof starts by repeating the argument made in the unconstrained case. The combination of (H1) and (HS1) allows us to claim that there is no loss of generality in restricting the minimization to $b_1 = \cdots = b_{m-1} = 0$. Then, given any minimizing sequence $v_q^{(\ell)} \in \mathcal{H}_q$, $q = 1, \ldots, m$, $M_q^{(\ell)} \in \mathcal{M}_{d_{q+1}, d_q}(\mathbb{R})$, $b_q^{(\ell)} \in \mathbb{R}^{d_{q+1}}$, $q = 0, \ldots, m$, with $b_0^{(\ell)} = \cdots = b_{m-1}^{(\ell)} = 0$, one can find a subsequence such that each $v_q^{(\ell)}$ converges weakly to $v_q^{(*)} \in \mathcal{H}_q$, and $M_q^{(\ell)}, b_q^{(\ell)}$

converge to $M_q^{(*)}, b_q^{(*)}$, and such that the limit achieves the minimum of the objective function in (5), with the additional property that $\boldsymbol{\varphi}_{v_q^{(\ell)}}$ converges uniformly to $\boldsymbol{\varphi}_{v_q^{(*)}}$ (which also ensures that the sequence $\zeta^{q(\ell)}$ converges to a limit $\zeta^{q(*)}$).

The only point that remains to be shown in the sub-Riemannian context is that $v_q^{(*)}$ satisfies the constraints, i.e., that $v_q^{(*)} \in \mathcal{W}_q(\boldsymbol{\varphi}_{v_q^{(*)}}(\cdot), \zeta^{q(*)})$, $q = 1, \ldots, m$. We now proceed with the argument.

Given a continuous function $\boldsymbol{\varphi} : t \mapsto \boldsymbol{\varphi}(t)$ and $\zeta \in (\mathbb{R}^{d_q})^n$, let $\boldsymbol{P}_{q,\boldsymbol{\varphi},\zeta}$ be defined on $\mathcal{H}_q$ by $\boldsymbol{P}_{q,\boldsymbol{\varphi},\zeta}(v)(t) = P_{W_q(\boldsymbol{\varphi}(t),\zeta)}(v(t))$. Clearly, $\boldsymbol{P}_{q,\boldsymbol{\varphi},\zeta}$ is bounded, maps $\mathcal{H}_q$ to $\mathcal{W}_q(\boldsymbol{\varphi}(\cdot), \zeta)$, and $\boldsymbol{P}_{q,\boldsymbol{\varphi},\zeta}(v) = v$ if and only if $v \in \mathcal{W}_q(\boldsymbol{\varphi}(\cdot), \zeta)$, showing that this set is closed and that $\boldsymbol{P}_{q,\boldsymbol{\varphi},\zeta}$ is its orthogonal projection. Moreover, if $\boldsymbol{\varphi}^{(\ell)}$ converges to $\boldsymbol{\varphi}$ and $\zeta^{(\ell)}$ to ζ, then $\boldsymbol{P}_{q,\boldsymbol{\varphi}^{(\ell)},\zeta^{(\ell)}}$ converges to $\boldsymbol{P}_{q,\boldsymbol{\varphi},\zeta}$, as can be deduced by dominated convergence and the hypotheses made on $P_{W_q(\boldsymbol{\varphi}(t),\zeta)}$.

Returning to $v_q^{(*)}$, assume that $v \in \mathcal{H}_q$ is perpendicular to $\mathcal{W}_q(\boldsymbol{\varphi}_{v_q^{(*)}}(\cdot), \zeta^{q(*)})$. Then

$$|\langle v, v_q^{(\ell)}\rangle_{\mathcal{H}_q}| = |\langle \boldsymbol{P}_{q,\boldsymbol{\varphi}_{v_q^{(\ell)}},\zeta^{q(\ell)}}(v), v_q^{(\ell)}\rangle_{\mathcal{H}_q}| \leq \|\boldsymbol{P}_{q,\boldsymbol{\varphi}_{v_q^{(\ell)}},\zeta^{q(\ell)}}(v)\|_{\mathcal{H}_q}\|v_q^{(\ell)}\|_{\mathcal{H}_q}.$$

Since $\boldsymbol{P}_{q,\boldsymbol{\varphi}_{v_q^{(\ell)}},\zeta^{q(\ell)}}(v)$ converges to $\boldsymbol{P}_{q,\boldsymbol{\varphi}_{v_q^{(*)}},\zeta^{q(*)}}(v) = 0$, we find that $\langle v, v_q^{(\ell)}\rangle_{\mathcal{H}_q}$ tends to 0. By weak convergence, this quantity also converges to $\langle v, v_q^{(*)}\rangle_{\mathcal{H}_q}$, which must therefore also vanish. Since this is true for all $v \in \mathcal{W}_q(\boldsymbol{\varphi}_{v_q^{(*)}}(\cdot), \zeta^{q(*)})^\perp$, we find that $v_q^{(*)} \in (\mathcal{W}_q(\boldsymbol{\varphi}_{v_q^{(*)}}(\cdot), \zeta^{q(*)})^\perp)^\perp = \mathcal{W}_q(\boldsymbol{\varphi}_{v_q^{(*)}}(\cdot), \zeta^{q(*)})$ (since the space is closed). This concludes the proof in the sub-Riemannian case.

To conclude, we check that (HS1) and (HS2) hold in the context of Section 8 (assuming that (H1) is true). In that section, the finite-dimensional space is generated by the columns of the matrices $K_q(\cdot, z_l)$, $l = 1, \ldots, n_S$, which leads us to define

$$W_q(\psi, \zeta) = \left\{ \sum_{l=1}^{n_S} K_q(\cdot, z_l) w_l : w_1, \ldots, w_{n_S} \in \mathbb{R}^{d_q}, z_l = \psi(\zeta_l) \right\},$$

for a diffeomorphism ψ and $\zeta \in (\mathbb{R}^{d_q})^n$. We have $f \in b \cdot W_q(\psi, \zeta)$ if and only if there exists $w_1, \ldots, w_{n_S}$ such that, for all $x \in \mathbb{R}^{d_q}$,

$$f(x) = \sum_{l=1}^{n_S} K_q(x + b, z_l) w_l$$

with $z_l = \psi(\zeta_l)$. By translation invariance of the norm in V_q, this is equivalent to

$$f(x) = \sum_{l=1}^{n_S} K_q(x, z_l - b) w_l.$$

We have $z_l - b = \psi(\zeta_l - b + b) - b = (b \cdot \psi)(\zeta_l - b)$ showing that $f \in b \cdot W_q(\psi, \zeta)$ is equivalent to $f \in W_q(b \cdot \psi, \zeta - b)$, proving (HS1).

Continuity of the projections is true because $P_{W(\psi,\zeta^q)}(f)$ for $f \in V_q$ takes the form

$$\sum_{l=1}^{n_S} K_q(\cdot, z_l) w_l(f),$$

where $w_1(f), \ldots, w_{n_S}(f)$ satisfy the linear system

$$\sum_{l=1}^{n_S} K_q(z_k, z_l) w_l(f) = f(z_k), \quad k = 1, \ldots, n_S,$$

which has a unique solution, continuous in z (over the set of n_S distinct points in $\mathbb{R}^{d_q}$) and thus in ψ and ζ.

Appendix B. Necessary Conditions for Optimality

Recall the notation for the general case of sub-optimal vector fields in Section 8, where a subset of the training data of size $n_S \leq n$ is selected, the training data renumbered such that the first n_S elements coincide with this subset, and $(z_1^q(\cdot), \ldots, z_{n_S}^q(\cdot))$ and $\boldsymbol{a}^q(\cdot) = (a_1^q(\cdot), \ldots, a_{n_S}^q(\cdot))$ represent the states corresponding to this subset and the controls, respectively. (For the optimal vector fields case, $n_S = n$.) We now let G denote the general reduced objective function in (4), namely

$$G(\boldsymbol{a}^1(\cdot), \ldots, \boldsymbol{a}^m(\cdot), A_0, \ldots, A_m) = \sum_{q=1}^{m} \int_0^1 L_q(\boldsymbol{z}^q(t), \boldsymbol{a}^q(t)) dt + \lambda \sum_{q=0}^{m} U_q(A_q) + \frac{1}{\sigma^2} \sum_{k=1}^{n} \gamma_k,$$

where the Lagrangian or running cost functions are $L_q : (\mathbb{R}^{d_q})^{n_S} \times (\mathbb{R}^{d_q})^{n_S} \to \mathbb{R}$, with

$$L_q(\boldsymbol{u}, \boldsymbol{w}) = \sum_{k,l=1}^{n_S} w_k^\top K_q(u_k, u_l) w_l,$$

and

$$\gamma_k = \Gamma_k(\pi_r(\zeta_k^{m+1})).$$

The dynamical system constraints are, for $k = 1, \ldots, n$: $\xi_k^0 = \iota_s(x_k)$, $\zeta_k^q = A_{q-1}(\xi_k^{q-1})$, $z_k^q(0) = \zeta_k^q$, $\partial_t z_k^q(t) = v_q(t)(z_k^q(t))$, $\xi_k^q = z_k^q(1)$, where

$$v_q(t)(\cdot) = \sum_{l=1}^{n_S} K_q(\cdot, z_l^q(t)) a_l^q(t).$$

Adjoin these constraints to G by the Lagrange multipliers $\rho_1^q, \ldots, \rho_n^q \in \mathbb{R}^{d_q}$ and $\boldsymbol{p}^q(\cdot) = (p_1^q(\cdot), \ldots, p_n^q(\cdot))$, $q = 1, \ldots, m$,

$$\sum_{q=1}^{m} \left[\sum_{k=1}^{n} {\rho_k^q}^\top (z_k^q(0) - \zeta_k^q) + \int_0^1 \left(\sum_{k=1}^{n} {p_k^q}^\top \partial_t z_k^q - H_{\boldsymbol{a}^q}^q(\boldsymbol{z}^q, \boldsymbol{p}^q) \right) dt \right] + \lambda \sum_{q=0}^{m} U_q(A_q) + \frac{1}{\sigma^2} \sum_{k=1}^{n} \gamma_k,$$

where $p_k^q(\cdot)$ is a time-dependent vector in $\mathbb{R}^{d_q}$ and $H_{\boldsymbol{w}}^q : (\mathbb{R}^{d_q})^n \times (\mathbb{R}^{d_q})^n \to \mathbb{R}$, $\boldsymbol{w} \in (\mathbb{R}^{d_q})^{n_S}$,

$$H_{\boldsymbol{w}}^q(\boldsymbol{u}, \boldsymbol{r}) = \sum_{k=1}^{n} r_k^\top \sum_{l=1}^{n_S} K_q(u_k, u_l) w_l - L_q(\boldsymbol{u}_S, \boldsymbol{w})$$

are the Hamiltonians, with $\boldsymbol{u}_S = (u_i \in \boldsymbol{u}, i = \{1, \ldots, n_S\})$. Apply the calculus of variations:

$$\sum_{q=1}^{m} \Bigg[\sum_{k=1}^{n} ((z_k^q(0) - \zeta_k^q)\delta\rho_k^q + \rho_k^q \delta z_k^q|_0 - \rho_k^q \partial_{M_{q-1}} \zeta_k^q \delta M_{q-1} - \rho_k^q \partial_{b_{q-1}} \zeta_k^q \delta b_{q-1} - \rho_k^q \partial_{\xi_k^{q-1}} \zeta_k^q \delta \xi_k^{q-1})$$
$$+ \int_0^1 \Bigg(\sum_{k=1}^{n} (\partial_t z_k^q \delta p_k^q + p_k^q \partial_t \delta z_k^q - \partial_{p_k^q} H_{\boldsymbol{a}^q}^q \delta p_k^q - \partial_{z_k^q} H_{\boldsymbol{a}^q}^q \delta z_k^q) - \sum_{k=1}^{n_S} (\partial_{a_k^q} H_{\boldsymbol{a}^q}^q \delta a_k^q) \Bigg) dt \Bigg]$$
$$+ \lambda \sum_{q=0}^{m} (\partial_{M_q} U_q(A_q) \delta M_q + \partial_{b_q} U_q(A_q) \delta b_q) + \frac{1}{\sigma^2} \sum_{k=1}^{n} (\partial_{M_m} \gamma_k \delta M_m + \partial_{b_m} \gamma_k \delta b_m + \partial_{\xi_m^q} \gamma_k \delta \xi_m^q).$$

Substitute $\partial_{b_q} U_q(A_q) = 0 \in \mathbb{R}^{d_{q+1}}$ and

$$\int_0^1 p_k^q \partial_t \delta z_k^q dt = (p_k^q \delta z_k^q)|_1 - (p_k^q \delta z_k^q)|_0 - \int_0^1 \partial_t p_k^q \delta z_k^q dt = p_k^q(1) \delta \xi_k^q - (p_k^q \delta z_k^q)|_0 - \int_0^1 \dot{p}_k^q \delta z_k^q dt :$$

$$\sum_{q=1}^{m} \Bigg[\sum_{k=1}^{n} ((z_k^q(0) - \zeta_k^q)\delta\rho_k^q + (\rho_k^q \delta z_k^q - p_k^q \delta z_k^q)|_0$$
$$- \rho_k^q (\xi_k^{q-1})^\top \delta M_{q-1} - \rho_k^q \delta b_{q-1} - M_{q-1}^\top \rho_k^q \delta \xi_k^{q-1} + p_k^q(1) \delta \xi_k^q)$$
$$+ \int_0^1 \Bigg(\sum_{k=1}^{n} (\partial_t z_k^q \delta p_k^q - \partial_t p_k^q \delta z_k^q - \partial_{p_k^q} H_{\boldsymbol{a}^q}^q \delta p_k^q - \partial_{z_k^q} H_{\boldsymbol{a}^q}^q \delta z_k^q) - \sum_{k=1}^{n_S} (\partial_{a_k^q} H_{\boldsymbol{a}^q}^q \delta a_k^q) \Bigg) dt \Bigg]$$
$$+ \lambda \sum_{q=0}^{m} \partial_{M_q} U_q(A_q) \delta M_q + \frac{1}{\sigma^2} \sum_{k=1}^{n} \big(\iota_r(\nabla \gamma_k) \xi_k^{m\top} \delta M_m + \iota_r(\nabla \gamma_k) \delta b_m + M_m^\top \iota_r(\nabla \gamma_k) \delta \xi_k^m \big).$$

Group terms by variation:

$$\sum_{q=1}^{m} \Bigg[\sum_{k=1}^{n} ((z_k^q(0) - \zeta_k^q)\delta\rho_k^q + (\rho_k^q - p_k^q(0)) \delta z_k^q|_0)$$
$$+ \int_0^1 \Bigg(\sum_{k=1}^{n} ((\partial_t z_k^q - \partial_{p_k^q} H_{\boldsymbol{a}^q}^q) \delta p_k^q - (\partial_t p_k^q + \partial_{z_k^q} H_{\boldsymbol{a}^q}^q) \delta z_k^q) - \sum_{k=1}^{n_S} (\partial_{a_k^q} H_{\boldsymbol{a}^q}^q \delta a_k^q) \Bigg) dt \Bigg]$$
$$+ \sum_{q=1}^{m-1} \sum_{k=1}^{n} (p_k^q(1) - M_q^\top \rho_k^{q+1}) \delta \xi_k^q + \sum_{k=1}^{n} \Bigg(\frac{1}{\sigma^2} M_m^\top \iota_r(\nabla \gamma_k) + p_k^m(1) \Bigg) \delta \xi_k^m$$
$$+ \sum_{q=0}^{m-1} \Bigg(\lambda \partial_{M_q} U_q(A_q) - \sum_{k=1}^{n} \rho_k^{q+1} \xi_k^{q\top} \Bigg) \delta M_q - \sum_{q=0}^{m-1} \sum_{k=1}^{n} \rho_k^{q+1} \delta b_q$$
$$+ \Bigg(\frac{1}{\sigma^2} \sum_{k=1}^{n} \iota_r(\nabla \gamma_k) \xi_k^{m\top} + \lambda \partial_{M_m} U_m(A_m) \Bigg) \delta M_m + \Bigg(\frac{1}{\sigma^2} \sum_{k=1}^{n} \iota_r(\nabla \gamma_k) \Bigg) \delta b_m.$$

Setting the coefficients of the variations with respect to the forward states and boundary conditions to zero, we have the following backpropagation states and boundary conditions:

$$
\begin{aligned}
p_k^m(1) &= -\frac{1}{\sigma^2} M_m^\top \iota_r(\nabla \Gamma_k(\pi_r(\zeta_k^{m+1}))) \\
p_k^q(1) &= M_q^\top \rho_k^{q+1}, \quad q = m-1, \dots, 1
\end{aligned}
$$

which imply

$$
\begin{aligned}
\rho_k^{m+1} &= -\frac{1}{\sigma^2} \iota_r(\nabla \Gamma_k(\pi_r(\zeta_k^{m+1}))) \\
p_k^q(1) &= M_q^\top \rho_k^{q+1}, \quad q = m, \dots, 1,
\end{aligned}
$$

and

$$
\begin{aligned}
\partial_t p_k^q(t) &= -\partial_{z_k^q(t)} H_{\boldsymbol{a}^q}^q(\boldsymbol{z}^q, \boldsymbol{p}^q), \quad q = m, \dots, 1 \\
&= -\sum_{l=1}^{n_S} \nabla_1 K_q(z_k^q(t), z_l^q(t)) p_k^q(t)^\top a_l^q(t) \\
&\qquad - \begin{cases} \sum_{l=1}^{n} \nabla_1 K_q(z_k^q(t), z_l^q(t)) a_k^q(t)^\top p_l^q(t) & \\ \qquad -2\sum_{l=1}^{n_S} \nabla_1 K_q(z_k^q(t), z_l^q(t)) a_k^q(t)^\top a_l^q(t), & \text{if } k \le n_S \\ 0, & \text{if } k > n_S \end{cases} \\
\rho_k^q &= p_k^q(0), \quad q = m, \dots, 1.
\end{aligned}
$$

The coefficients of the variations with respect to our parameters are the gradients

$$
\begin{aligned}
\partial_{a_k^q(t)} G &= -\partial_{a_k^q(t)} H_{\boldsymbol{a}^q}^q(\boldsymbol{z}^q, \boldsymbol{p}^q), \quad k = 1, \dots, n_S, \quad q = 1, \dots, m \\
&= 2\sum_{l=1}^{n_S} K_q(z_k^q(t), z_l^q(t)) a_l^q(t) - \sum_{l=1}^{n} K_q(z_k^q(t), z_l^q(t)) p_l^q(t) \\
\partial_{M_q} G &= \lambda \partial_{M_q} U_q(A_q) - \sum_{k=1}^{n} \rho_k^{q+1} \xi_k^{q\top}, \quad q = 0, \dots, m \\
\partial_{b_q} G &= -\sum_{k=1}^{n} \rho_k^{q+1}, \quad q = 0, \dots, m,
\end{aligned}
$$

which are calculated using the backpropagation states.

By the PMP, our optimal controls $\boldsymbol{a}^q(\cdot)$ and state trajectories $\boldsymbol{z}^q(\cdot)$ must also solve these Hamiltonian systems with corresponding costates $\boldsymbol{p}^q(\cdot)$ and stationarity conditions

$$
\boldsymbol{a}^q(t) = \underset{\boldsymbol{a}'(t)}{\operatorname{argmax}} H_{\boldsymbol{a}'(t)}^q(\boldsymbol{z}^q(t), \boldsymbol{p}^q(t)).
$$

Therefore, an optimal minimizer of our learning problem sets the above gradients to zero.

Appendix C. Literature Results

	UCI Gap Splits				
Model	**Concrete**	**Energy**	**Kin8nm**	**Naval**	**Power**
MAP-1	7.79 ± 0.18	2.83 ± 0.99	0.09 ± 0.01	0.02 ± 0.00	4.24 ± 0.12
MAP-2	7.78 ± 0.23	3.70 ± 1.33	0.08 ± 0.00	0.03 ± 0.00	4.33 ± 0.18
MAP-1 NL	7.68 ± 0.23	3.09 ± 1.17	0.09 ± 0.01	0.02 ± 0.00	4.25 ± 0.09
MAP-2 NL	7.44 ± 0.17	3.48 ± 1.21	0.07 ± 0.00	0.03 ± 0.00	4.27 ± 0.08
Reg-1 NL	8.21 ± 0.48	4.24 ± 2.11	0.08 ± 0.00	0.01 ± 0.00	5.17 ± 0.60
Reg-2 NL	8.27 ± 0.39	3.83 ± 1.49	0.07 ± 0.00	0.01 ± 0.00	5.23 ± 0.43
BN(ML)-1 NL	7.69 ± 0.51	4.15 ± 1.64	0.09 ± 0.00	0.01 ± 0.00	4.49 ± 0.15
BN(ML)-2 NL	7.33 ± 0.36	4.10 ± 1.64	0.08 ± 0.00	0.01 ± 0.00	5.17 ± 0.28
BN(BO)-1 NL	7.74 ± 0.31	4.76 ± 1.98	0.08 ± 0.00	0.01 ± 0.00	4.66 ± 0.21
BN(BO)-2 NL	9.20 ± 0.55	4.58 ± 1.87	0.07 ± 0.00	0.01 ± 0.00	5.27 ± 0.36
DUN	7.20 ± 0.18	2.94 ± 0.67	0.08 ± 0.00	0.02 ± 0.00	4.30 ± 0.09
DUN (MLP)	7.46 ± 0.21	3.61 ± 0.88	0.08 ± 0.00	0.02 ± 0.00	4.58 ± 0.08
Dropout	7.06 ± 0.21	2.87 ± 0.50	0.07 ± 0.00	0.03 ± 0.00	4.69 ± 0.07
Ensemble	6.85 ± 0.18	3.36 ± 0.83	1.63 ± 0.99	0.02 ± 0.00	4.37 ± 0.09
MFVI	7.55 ± 0.19	8.61 ± 2.10	0.10 ± 0.01	0.03 ± 0.01	4.68 ± 0.16
SGD	7.37 ± 0.19	3.06 ± 0.64	0.09 ± 0.00	0.02 ± 0.00	4.62 ± 0.08

Table C.1A: Average test RMSE ± 1 standard error.

	UCI Gap Splits		
Model	**Protein**	**Wine**	**Yacht**
MAP-1	5.16 ± 0.04	0.63 ± 0.01	1.31 ± 0.14
MAP-2	5.07 ± 0.06	0.63 ± 0.01	1.05 ± 0.09
MAP-1 NL	5.13 ± 0.05	0.63 ± 0.01	1.28 ± 0.14
MAP-2 NL	5.08 ± 0.06	0.63 ± 0.01	1.01 ± 0.09
Reg-1 NL	5.23 ± 0.12	0.66 ± 0.02	1.24 ± 0.11
Reg-2 NL	5.33 ± 0.16	0.64 ± 0.01	1.22 ± 0.13
BN(ML)-1 NL	5.27 ± 0.12	0.63 ± 0.01	1.15 ± 0.11
BN(ML)-2 NL	5.37 ± 0.17	0.64 ± 0.01	1.31 ± 0.16
BN(BO)-1 NL	5.14 ± 0.10	0.65 ± 0.01	1.37 ± 0.15
BN(BO)-2 NL	5.46 ± 0.17	0.64 ± 0.01	1.59 ± 0.23
DUN	5.21 ± 0.35	0.70 ± 0.01	1.85 ± 0.17
DUN (MLP)	5.10 ± 0.24	0.69 ± 0.01	1.85 ± 0.14
Dropout	5.13 ± 0.28	0.66 ± 0.01	2.29 ± 0.47
Ensemble	4.80 ± 0.27	0.67 ± 0.01	1.84 ± 0.19
MFVI	5.12 ± 0.13	0.63 ± 0.01	1.84 ± 0.16
SGD	5.17 ± 0.28	0.73 ± 0.02	2.21 ± 0.18

Table C.1B: Average test RMSE ± 1 standard error.

	UCI Standard Splits (Different Splits in Gray)				
Model	**Concrete**	**Energy**	**Kin8nm**	**Naval**	**Power**
VI	7.13 ± 0.12	2.65 ± 0.08	0.10 ± 0.00	0.01 ± 0.00	4.33 ± 0.04
BP	5.98 ± 0.22	1.10 ± 0.07	0.09 ± 0.00	0.00 ± 0.00	4.18 ± 0.04
BP-2	5.40 ± 0.13	0.68 ± 0.04	0.07 ± 0.00	0.00 ± 0.00	4.22 ± 0.07
BP-3	5.57 ± 0.13	0.63 ± 0.03	0.07 ± 0.00	0.00 ± 0.00	4.11 ± 0.04
BP-4	5.53 ± 0.14	0.67 ± 0.03	0.07 ± 0.00	0.00 ± 0.00	4.18 ± 0.06
PBP	5.67 ± 0.09	1.80 ± 0.05	0.10 ± 0.00	0.01 ± 0.00	4.12 ± 0.03
PBP-2	5.24 ± 0.12	0.90 ± 0.05	0.07 ± 0.00	0.00 ± 0.00	4.03 ± 0.03
PBP-3	5.73 ± 0.11	1.24 ± 0.06	0.07 ± 0.00	0.01 ± 0.00	4.07 ± 0.04
PBP-4	5.96 ± 0.16	1.18 ± 0.06	0.08 ± 0.00	0.00 ± 0.00	4.08 ± 0.04
Dropout-TS	5.23 ± 0.12	1.66 ± 0.04	0.10 ± 0.00	0.01 ± 0.00	4.02 ± 0.04
VMG	4.70 ± 0.14	1.16 ± 0.03	0.08 ± 0.00	0.00 ± 0.00	3.88 ± 0.03
HS-BNN	5.66 ± 0.09	1.99 ± 0.08	0.08 ± 0.00	0.00 ± 0.00	4.03 ± 0.03
PBP-MV	5.08 ± 0.14	0.45 ± 0.01	0.07 ± 0.00	0.00 ± 0.00	3.91 ± 0.04
Dropout-C	4.93 ± 0.14	1.08 ± 0.03	0.09 ± 0.00	0.00 ± 0.00	4.00 ± 0.04
Dropout-G	4.82 ± 0.16	0.54 ± 0.06	0.08 ± 0.00	0.00 ± 0.00	4.01 ± 0.04
BBB	6.16 ± 0.13	0.97 ± 0.09	0.08 ± 0.00	0.00 ± 0.00	4.21 ± 0.03
SLANG	5.58 ± 0.19	0.64 ± 0.03	0.08 ± 0.00	0.00 ± 0.00	4.16 ± 0.04
MAP-1	5.41 ± 0.12	0.52 ± 0.02	0.08 ± 0.00	0.00 ± 0.00	4.11 ± 0.04
MAP-2	5.13 ± 0.12	0.47 ± 0.02	0.07 ± 0.00	0.00 ± 0.00	3.99 ± 0.03
MAP-1 NL	5.14 ± 0.13	0.44 ± 0.01	0.08 ± 0.00	0.00 ± 0.00	4.01 ± 0.04
MAP-2 NL	5.05 ± 0.11	0.42 ± 0.02	0.07 ± 0.00	0.00 ± 0.00	3.90 ± 0.04
Reg-1 NL	5.03 ± 0.16	0.46 ± 0.01	0.08 ± 0.00	0.00 ± 0.00	3.91 ± 0.04
Reg-2 NL	4.82 ± 0.14	0.43 ± 0.02	0.07 ± 0.00	0.00 ± 0.00	3.74 ± 0.04
BN(ML)-1 NL	5.08 ± 0.13	0.46 ± 0.01	0.08 ± 0.00	0.00 ± 0.00	3.94 ± 0.04
BN(ML)-2 NL	5.17 ± 0.12	0.42 ± 0.01	0.07 ± 0.00	0.00 ± 0.00	3.73 ± 0.04
BN(BO)-1 NL	4.96 ± 0.15	0.48 ± 0.01	0.08 ± 0.00	0.00 ± 0.00	3.94 ± 0.04
BN(BO)-2 NL	4.78 ± 0.19	0.40 ± 0.01	0.07 ± 0.00	0.00 ± 0.00	3.70 ± 0.04
DUN	4.61 ± 0.14	0.61 ± 0.04	0.08 ± 0.00	0.00 ± 0.00	3.57 ± 0.06
DUN (MLP)	4.57 ± 0.16	0.95 ± 0.11	0.08 ± 0.00	0.00 ± 0.00	3.67 ± 0.06
Dropout	4.61 ± 0.13	0.57 ± 0.05	0.07 ± 0.00	0.00 ± 0.00	3.82 ± 0.08
Ensemble	4.55 ± 0.13	0.51 ± 0.02	0.30 ± 0.22	0.00 ± 0.00	3.44 ± 0.05
MFVI	5.89 ± 0.17	1.69 ± 0.23	0.08 ± 0.00	0.01 ± 0.00	4.29 ± 0.04
SGD	4.98 ± 0.20	0.80 ± 0.06	0.20 ± 0.12	0.00 ± 0.00	3.70 ± 0.06
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{0})$	6.08 ± 0.15	2.25 ± 0.08	0.09 ± 0.00	0.00 ± 0.00	4.06 ± 0.04
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{0.25})$	5.79 ± 0.17	1.81 ± 0.07	0.08 ± 0.00	0.00 ± 0.00	4.04 ± 0.04
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{0.5})$	5.61 ± 0.15	1.12 ± 0.06	0.08 ± 0.00	0.00 ± 0.00	4.04 ± 0.04
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{0.75})$	5.67 ± 0.16	1.31 ± 0.10	0.08 ± 0.00	0.00 ± 0.00	4.04 ± 0.03
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{1.0})$	5.55 ± 0.17	1.54 ± 0.12	0.08 ± 0.00	0.00 ± 0.00	4.06 ± 0.04
$\mathcal{L}_{\mathbf{MM}}$	6.28 ± 0.18	2.19 ± 0.06	0.08 ± 0.00	0.00 ± 0.00	4.07 ± 0.04
$\mathcal{L}_{\mathbf{MSE}}$	4.96 ± 0.14	0.92 ± 0.02	0.08 ± 0.00	0.00 ± 0.00	4.01 ± 0.04
Student-t	5.82 ± 0.13	2.26 ± 0.08	0.09 ± 0.00	0.00 ± 0.00	4.02 ± 0.04
xVAMP	5.44 ± 0.14	1.87 ± 0.07	0.08 ± 0.00	0.00 ± 0.00	4.03 ± 0.04
xVAMP*	5.35 ± 0.16	2.00 ± 0.06	0.08 ± 0.00	0.00 ± 0.00	4.03 ± 0.04
VBEM	5.21 ± 0.13	1.29 ± 0.07	0.08 ± 0.00	0.00 ± 0.00	4.09 ± 0.03
VBEM*	5.17 ± 0.13	1.08 ± 0.04	0.08 ± 0.00	0.00 ± 0.00	4.02 ± 0.04

Table C.2A: Average test RMSE $\pm$ 1 standard error.

	UCI Standard Splits (Different Splits in Gray)			
Model	**Protein**	**Wine**	**Yacht**	**Year**
VI	4.84 ± 0.03	0.65 ± 0.01	6.89 ± 0.67	$9.03 \pm \mathrm{NA}$
BP	4.54 ± 0.03	0.65 ± 0.01	1.18 ± 0.16	$8.93 \pm \mathrm{NA}$
BP-2	4.19 ± 0.03	0.65 ± 0.01	1.54 ± 0.19	$8.98 \pm \mathrm{NA}$
BP-3	4.01 ± 0.03	0.65 ± 0.01	1.11 ± 0.09	$8.93 \pm \mathrm{NA}$
BP-4	3.96 ± 0.01	0.65 ± 0.02	1.27 ± 0.13	$9.05 \pm \mathrm{NA}$
PBP	4.73 ± 0.01	0.64 ± 0.01	1.02 ± 0.05	$8.88 \pm \mathrm{NA}$
PBP-2	4.25 ± 0.02	0.64 ± 0.01	0.85 ± 0.05	$8.92 \pm \mathrm{NA}$
PBP-3	4.09 ± 0.03	0.64 ± 0.01	0.89 ± 0.10	$8.87 \pm \mathrm{NA}$
PBP-4	3.97 ± 0.04	0.64 ± 0.01	1.71 ± 0.23	$8.93 \pm \mathrm{NA}$
Dropout-TS	4.36 ± 0.01	0.62 ± 0.01	1.11 ± 0.09	$8.85 \pm \mathrm{NA}$
VMG	4.14 ± 0.01	0.61 ± 0.01	0.77 ± 0.06	$8.78 \pm \mathrm{NA}$
HS-BNN	4.39 ± 0.02	0.63 ± 0.01	1.58 ± 0.05	$9.26 \pm \mathrm{NA}$
PBP-MV	3.94 ± 0.02	0.64 ± 0.01	0.81 ± 0.06	$8.72 \pm \mathrm{NA}$
Dropout-C	4.27 ± 0.01	0.61 ± 0.01	0.70 ± 0.05	$--$
Dropout-G	4.27 ± 0.02	0.62 ± 0.01	0.67 ± 0.05	$--$
BBB	$--$	0.64 ± 0.01	1.13 ± 0.06	$--$
SLANG	$--$	0.65 ± 0.01	1.08 ± 0.06	$--$
MAP-1	4.67 ± 0.03	0.64 ± 0.01	0.73 ± 0.06	$--$
MAP-2	4.33 ± 0.01	0.63 ± 0.01	0.66 ± 0.06	$--$
MAP-1 NL	4.56 ± 0.01	0.64 ± 0.01	0.61 ± 0.05	$--$
MAP-2 NL	4.24 ± 0.01	0.63 ± 0.01	0.63 ± 0.05	$--$
Reg-1 NL	4.25 ± 0.02	0.64 ± 0.01	0.64 ± 0.04	$--$
Reg-2 NL	3.94 ± 0.02	0.63 ± 0.01	0.58 ± 0.06	$--$
BN(ML)-1 NL	4.24 ± 0.01	0.63 ± 0.01	0.79 ± 0.06	$--$
BN(ML)-2 NL	3.94 ± 0.02	0.63 ± 0.01	0.55 ± 0.05	$--$
BN(BO)-1 NL	4.25 ± 0.01	0.63 ± 0.01	0.77 ± 0.06	$--$
BN(BO)-2 NL	3.88 ± 0.02	0.63 ± 0.01	0.66 ± 0.06	$--$
DUN	3.40 ± 0.03	0.66 ± 0.01	2.51 ± 0.44	$--$
DUN (MLP)	3.41 ± 0.03	0.63 ± 0.01	2.47 ± 0.19	$--$
Dropout	3.43 ± 0.03	0.64 ± 0.01	0.88 ± 0.09	$--$
Ensemble	3.26 ± 0.03	1.93 ± 1.28	1.43 ± 0.11	$--$
MFVI	4.51 ± 0.06	0.66 ± 0.01	3.42 ± 1.64	$--$
SGD	3.59 ± 0.08	0.65 ± 0.01	2.35 ± 0.20	$--$
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{0})$	4.49 ± 0.05	0.64 ± 0.01	1.22 ± 0.11	$--$
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{0.25})$	4.35 ± 0.02	0.64 ± 0.01	1.73 ± 0.22	$--$
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{0.5})$	4.31 ± 0.01	0.64 ± 0.01	2.35 ± 0.32	$--$
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{0.75})$	4.28 ± 0.01	0.64 ± 0.01	1.97 ± 0.23	$--$
$\mathcal{L}_{\beta-\mathbf{NLL}}(\beta = \mathbf{1.0})$	4.31 ± 0.02	0.64 ± 0.01	2.08 ± 0.25	$--$
$\mathcal{L}_{\mathbf{MM}}$	4.32 ± 0.03	0.65 ± 0.01	3.02 ± 0.31	$--$
$\mathcal{L}_{\mathbf{MSE}}$	4.28 ± 0.03	0.63 ± 0.01	0.78 ± 0.06	$--$
Student-t	4.76 ± 0.11	0.64 ± 0.01	1.34 ± 0.14	$--$
xVAMP	4.38 ± 0.02	0.64 ± 0.01	0.99 ± 0.10	$--$
xVAMP*	4.31 ± 0.01	0.63 ± 0.01	1.13 ± 0.15	$--$
VBEM	4.31 ± 0.00	0.64 ± 0.01	1.66 ± 0.19	$--$
VBEM*	4.35 ± 0.04	0.63 ± 0.01	0.65 ± 0.04	$--$

Table C.2B: Average test RMSE $\pm$ 1 standard error.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2015. URL `http://download.tensorflow.org/paper/whitepaper2015.pdf`.

B. Amor, S. Arguillere, and L. Shao. ResNet-LDDMM: Advancing the LDDMM framework using deep residual networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(03):3707–3720, 2023.

J. Antoran, J. Allingham, and J. Hernández-Lobato. Depth uncertainty in neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33*, pages 10620–10634. Curran Associates, Inc., 2020.

S. Arguillere, E. Trélat, A. Trouvé, and L. Younes. Shape deformation analysis from the optimal control viewpoint. *Journal de Mathématiques Pures et Appliquées*, 104(1):139–178, 2015.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

M. Beg, M. Miller, A. Trouvé, and L. Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International Journal of Computer Vision*, 61:139–157, 2005.

N. Boumal. *An Introduction to Optimization on Smooth Manifolds.* Cambridge University Press, 2023.

B. Charlier, J. Feydy, J. Glaunes, F. Collin, and G. Durif. Kernel operations on the GPU, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(1): 3457–3462, 2021.

R. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

E. Dupont, A. Doucet, and Y. Teh. Augmented neural ODEs. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3140–3150. Curran Associates, Inc., 2019.

A. Foong, Y. Li, J. Hernández-Lobato, and R. Turner. 'In-between' uncertainty in Bayesian neural networks. *arXiv:abs/1906.11537*, 2019.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In M. Balcan and K. Weinberger, editors, *International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059. PMLR, 2016.

N. Ganaba. Deep learning: Hydrodynamics, and Lie-Poisson Hamilton-Jacobi theory. *arXiv:2105.09542*, 2021.

S. Ghosh, J. Yao, and F. Doshi-Velez. Model selection in Bayesian neural networks via horseshoe priors. *Journal of Machine Learning Research*, 20(182):1–46, 2019. (First appeared in *arXiv:1705.10388*, 2017).

B. Gris, S. Durrleman, and A. Trouvé. A sub-Riemannian modular framework for diffeomorphism-based analysis of shape ensembles. *SIAM Journal on Imaging Sciences*, 11(1):802–833, 2018.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016.

J. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In F. Bach and D. Blei, editors, *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1861–1869. PMLR, 2015.

L. Hocking. *Optimal Control: An Introduction to the Theory with Applications.* Oxford University Press, 1991.

E. Jansson and K. Modin. Sub-Riemannian landmark matching and its interpretation as residual neural networks. *arXiv:2204.09351*, 2022.

S. Joshi and M. Miller. Landmark matching via large deformation diffeomorphisms. *IEEE Transactions in Image Processing*, 9(8):1357–1370, 2000.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

I. Kobyzev, S. Prince, and M. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 43 (11):3964–3979, 2021.

C. Louizos and M. Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In M. Balcan and K. Weinberger, editors, *International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1708–1716. PMLR, 2016.

J. Macki and A. Strauss. *Introduction to Optimal Control Theory.* Springer Science & Business Media, 2012.

C. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17 (1):177–204, 2005.

M. Miller, A. Trouvé, and L. Younes. On the metrics and Euler-Lagrange equations of computational anatomy. *Annual Review of Biomedical Engineering*, 4(1):375–405, 2002.

M. Miller, A. Trouvé, and L. Younes. Hamiltonian systems and optimal control in computational anatomy: 100 years since D'Arcy Thompson. *Annual Review of Biomedical Engineering*, 17(1):447–509, 2015.

A. Mishkin, F. Kunstner, D. Nielsen, M. Schmidt, and M. Khan. SLANG: Fast structured covariance approximations for Bayesian deep learning with natural gradient. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6248–6258. Curran Associates, Inc., 2018.

J. Mukhoti, P. Stenetorp, and Y. Gal. On the importance of strong baselines in Bayesian deep learning. *arXiv:1811.09385*, 2018.

S. Ober and C. Rasmussen. Benchmarking the neural linear model for regression. *arXiv:1912.08416*, 2019.

H. Owhadi. Do ideas have shape? Idea registration as the continuous limit of artificial neural networks. *Physica D: Nonlinear Phenomena*, 444:133592, 2023.

A. Queiruga, N. Erichson, D. Taylor, and M. Mahoney. Continuous-in-depth neural networks. *arXiv:2008.02389*, 2020.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In F. Bach and D. Blei, editors, *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538. PMLR, 2015.

F. Rousseau, L. Drumetz, and R. Fablet. Residual networks as flows of diffeomorphisms. *Journal of Mathematical Imaging and Vision*, 62:1–11, 2019.

L. Ruthotto. Differential equations for continuous-time deep learning. *Notices of the American Mathematical Society*, 71(5):613–620, 2024.

M. Seitzer, A. Tavakoli, D. Antic, and G. Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. *arXiv:2203.09168*, 2022.

S. Sun, C. Chen, and L. Carin. Learning structured weight uncertainty in Bayesian neural networks. In A. Singh and J. Zhu, editors, *International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1283–1292. PMLR, 2017.

M. Vaillant, M. Miller, L. Younes, and A. Trouvé. Statistics on diffeomorphisms via tangent space representations. *NeuroImage*, 23:S161–S169, 2004.

F. Vialard, R. Kwitt, S. Wei, and M. Niethammer. A shooting formulation of deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33*, pages 11828–11838. Curran Associates, Inc., 2020.

G. Wahba. *Spline Models for Observational Data*. SIAM, 1990.

C. Walder and B. Schölkopf. Diffeomorphic dimensionality reduction. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1713–1720. Curran Associates, Inc., 2009.

E. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 1(5):1–11, 2017.

N. Wu and M. Zhang. NeurEPDiff: Neural operators to predict geodesics in deformation spaces. *arXiv:2303.07115*, 2023.

L. Younes. *Shapes and Diffeomorphisms*. Springer, 2010. (Second edition: 2019).

L. Younes. Constrained diffeomorphic shape evolution. *Foundations of Computational Mathematics*, 12(3):295–325, 2012.

L. Younes. Diffeomorphic learning. *Journal of Machine Learning Research*, 21(1):9077–9104, 2020. (First appeared in *arXiv:1806.01240*, 2018).

L. Younes, B. Gris, and A. Trouvé. Sub-Riemannian methods in shape analysis. In *Handbook of Variational Methods for Nonlinear Geometric Data*, pages 463–495. Springer, 2020.