Neural ODEs with stochastic vector field mixtures

Niall Twomey, Michał Kozłowski, Raúl Santos-Rodríguez
University of Bristol
Bristol, United Kingdom
{niall.twomey, m.kozlowski, enrsr}@bristol.ac.uk

Abstract

It was recently shown that neural ordinary differential equation models cannot solve fundamental and seemingly straightforward tasks even with high-capacity vector field representations. This paper introduces two other fundamental tasks to the set that baseline methods cannot solve, and proposes mixtures of stochastic vector fields as a model class that is capable of solving these essential problems. Dynamic vector field selection is of critical importance for our model, and our approach is to propagate component uncertainty over the integration interval with a technique based on forward filtering. We also formalise several loss functions that encourage desirable properties on the trajectory paths, and of particular interest are those that directly encourage fewer expected function evaluations. Experimentally, we demonstrate that our model class is capable of capturing the natural dynamics of human behaviour; a notoriously volatile application area. Baseline approaches cannot adequately model this problem.

1 Introduction

Recent work has linked Residual Networks (ResNets) [1] to Ordinary Differential Equations (ODEs) and demonstrated that ResNets may be interpreted as Euler solutions to Initial Value Problems (IVPs) [2–4]. This idea has been further developed with ResNets taken to their continuous limit by Chen et al. [5] with the introduction of Neural ODEs (NODEs). In their setting, Neural Networks (NNs) produce Vector Field (VF) representations that are utilised by ODE solvers to learn and calculate explicit flow trajectories through the feature space. The resulting model can be seen to be of continuous depth since the ODE solver will query the VFs at arbitrary levels that are unspecified in advance. Their work also demonstrates efficient and scalable inference with the adjoint trick.

Vector fields are defined as follows in the NODE paradigm

$$\nabla \mathbf{h}(t_i) = f(\mathbf{h}(t_i), t_i; \boldsymbol{\theta}) \tag{1}$$

where $\mathbf{h}(t_i) \in \mathbb{R}^D$ is a hidden state at depth t_i (base state $\mathbf{h}(t_0) \triangleq \mathbf{x}$, maximal depth t_T), f is a neural network of arbitrary architecture and $\boldsymbol{\theta}$ are its parameters. Subsequent hidden states are derived by taking small steps from $\mathbf{h}(t_i)$ in the direction of the vector field, *i.e.* $\mathbf{h}(t_{i+1}) = \mathbf{h}(t_i) + \delta_{t_i} \nabla \mathbf{h}(t_i)$, and the step size δ_{t_i} is often selected automatically by an ODE solver. The solution of this IVP is obtained by repeatedly taking these steps until the output state $\mathbf{h}(t_T)$ is reached, and these outputs may undergo a final transformation, *e.g.* through a softmax layer in classification tasks. The optimisation objective is to adjust the dynamics of the ODE through $\boldsymbol{\theta}$ to maximise the data likelihood.

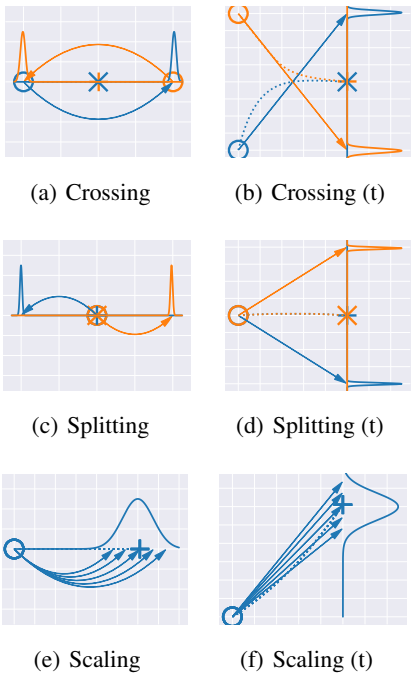
A critical component of NODEs is the parametrisation of the VFs and, since f is a neural network, the practitioner has the important responsibility of appropriately specifying its architecture. Even though the VF representations will be non-linear (and arbitrarily 'flexible' depend-

Preprint. Under review.

ing on architecture choice), NODE models cannot model arbitrary problems. This was clearly demonstrated by Dupont et al. [6] with a seemingly straightforward task in one dimension. A simple and elegant extension called Augmented NODEs (ANODEs) is proposed. These are experimentally shown to be a more expressive and stable model choice capable of solving the crossing problem.

Figure 1 illustrates NODE failure cases. Crossing [6] tasks cannot be solved by NODEs since their advections constitute homomorphisms for which crossing is forbidden. However, ANODE models find solutions by lifting the representation out of the original feature domain. Additionally, neither NODE nor ANODE models can solve splitting problems (where a single datapoint maps to more than one target) or account for uncertainty and hence they always map to a single target regardless of the variance of the system (*c.f.* scaling cases in the figure).

The shortcomings of baseline approaches are addressed with a probabilistic approach to ODE problems in this work. We use indoor localisation as a motivating application which seeks to find a mapping to a relative coordinate system. We propose the use of NODE-like models to learn typical patterns of movement and later to deploy these to forecast future behaviour. The three cases in fig. 1 are ofter encountered in behavioural modelling as follows: crossing: residents will never take the exact same path between two locations, and exemplar paths are likely to cross at least once; splitting: residents have the option to turn left, right, move forward or backtrack on their trajectory with their latent intent behind their motion determining this decision; and scaling: unobservable factors influence paths walked by residents, and even under identical starting conditions a resident may move at different speeds/directions along the same path.



(a) Crossing (b) Crossing (t) (c) Splitting (d) Splitting (t) (e) Scaling (f) Scaling (t)

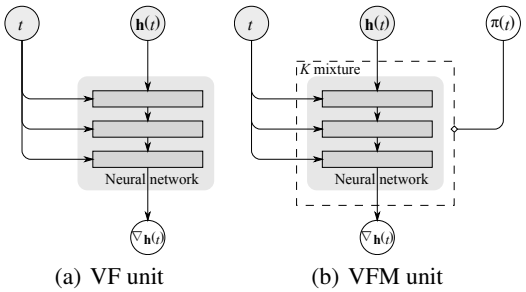
Figure 1: This figure illustrates failure cases for NODE models in their original space (left) and particle evolution over the integration interval (right). In these figures colour indicates class membership, the solid arrows link the start ($\circ$) and end ($\times$ +) positions of particles (*i.e.* ground truth data) and the dotted lines depict the learnt transportation by NODE models. The Gaussian PDFs represent the predictive distributions of the proposed method.

Our main contribution is the introduction and exposition of Stochastic VF Mixtures (SVFMs) within the NODE paradigm, and the introduction of trajectory-focused losses that encourage simpler and less distorted solutions. Their utility is primarily enabled by the uncertainty that is posited on both the vector fields themselves and on selection of vector fields from an ensemble, but is also supported by loss functions that encourage simpler solutions to the IVP and capture richer context in forecasting tasks. We demonstrate their capability in solving the above three fundamental tasks where baseline models fail. Although the proposed methods are developed specifically with forecasting in mind, we also demonstrate their utility in classical classification tasks.

2 Proposed methods

2.1 Vector field mixtures

The graphical model representing the basic VF unit is shown in fig. 2(a). The input variables (t and $\mathbf{h}(t)$) are denoted by shaded nodes indicating that these values are observed, whereas the output node ($\nabla\mathbf{h}(t)$) is unshaded, indicating that it is latent. The assumption here is that the i-th VF is conditionally independent of its history given its hidden state. In what follows we use the notation $f_g(\mathbf{h}(t), t; \boldsymbol{\theta}^{(g)})$ to represent a NODE



(a) VF unit (b) VFM unit

Figure 2: Architecture of VF and VFM units.

unit on an example latent variable g, whereas $f^{(k)}(\cdot)$ specifically defines the k-th VF from the VF Mixture (VFM).

Using gate notation [7], a VFM with K components is shown in fig. 2(b). The new latent variable, $\boldsymbol{\pi}(t) \in \triangle^K$ (where $\triangle^K$ defines the K-simplex) is a discrete probability distribution over component membership. The manner in which this distribution is specified is a key consideration with these models. Two approaches for defining $\boldsymbol{\pi}(t)$ and propagating uncertainty over the integration interval are outlined below.

Pick and stick. This operates under the assumption that component membership is constant over the forecast horizon, *i.e.* $\boldsymbol{\pi}(t_i) = \boldsymbol{\pi}(t_*) \ \forall\, i \ (0 \leq i \leq T)$. Several choices for specifying t_* may be considered, but if mixture dynamics are well specified at t_0 we can set

$$\boldsymbol{\pi}(t_*) = \boldsymbol{\pi}(t_0) = f_{\boldsymbol{\pi}_{t_0}}(\mathbf{h}(t_0), t_0; \boldsymbol{\theta}^{(\boldsymbol{\pi}_{t_0})}) \tag{2}$$

Forward filtering. Forward filtering is a technique used in dynamic systems to estimate belief based on a history of evidence, and it is often used in dynamic models like Hidden Markov Models (HMMs) [8] and Conditional Random Fields (CRFs) [9]. It is incorporated into our model directly by modeling transition and emission dynamics of component membership as follows

$$\boldsymbol{\Psi}(t_i) = f_{\boldsymbol{\Psi}}(\mathbf{h}(t_i), t_i; \boldsymbol{\theta}^{(\boldsymbol{\Psi})}) \tag{3}$$

$$\boldsymbol{\psi}(t_i) = f_{\boldsymbol{\psi}}(\mathbf{h}(t_i), t_i; \boldsymbol{\theta}^{(\boldsymbol{\psi})}) \tag{4}$$

where each of the K rows of $\boldsymbol{\Psi}(t_i)$ and $\boldsymbol{\psi}(t_i) \in \triangle^K$. Forward filtering aggregates belief as follows

$$\boldsymbol{\pi}(t_i) \propto \boldsymbol{\Psi}(t_i)^\top \left(\boldsymbol{\psi}\left(t_i\right) \odot \boldsymbol{\pi}\left(t_{i-1}\right)\right) \tag{5}$$

where $\odot$ depicts the Hadamard product, $\top$ the matrix transpose, and equality is achieved with normalisation. A complementary interpretation of forward filtering is that of a Recurrent NN (RNN) [10] that grows probabilistically through the forecasting horizon.

2.2 Stochastic vector fields

Uncertainty is introduced to VFs with the mean ($\boldsymbol{\mu}(t) = g_{\boldsymbol{\mu}}(f_{\mathbf{z}}(t), t; \boldsymbol{\theta}^{(\boldsymbol{\mu})})$) and variance ($\boldsymbol{\tau}(t) = g_{\boldsymbol{\tau}}(f_{\mathbf{z}}(t), t; \boldsymbol{\theta}^{(\boldsymbol{\tau})})$) generating functions that both arise from a common representation $f_{\mathbf{z}}(t) = g_z(\mathbf{h}(t), t; \boldsymbol{\theta}^{(z)})$, see fig. 3. It will be convenient to decompose VFs into the length ($v(t)$) and orientation ($\mathbf{u}(t)$) of the VF and to impose uncertainty on these separately:

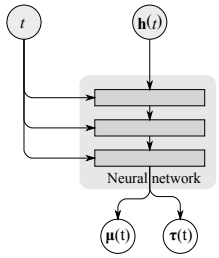


Figure 3: SVF unit.

$$\mathbf{u}(t) \sim \mathcal{N}_s\left(\boldsymbol{\mu}^{(\mathbf{u})}(t), \boldsymbol{\tau}^{(\mathbf{u})}(t)\right) \tag{6}$$

$$\log v(t) \sim \mathcal{N}\left(\log \boldsymbol{\mu}^{(v)}(t), \boldsymbol{\tau}^{(v)}(t)\right) \tag{7}$$

where $\boldsymbol{\mu}^{(\mathbf{u})}(\cdot)$, $\boldsymbol{\tau}^{(\mathbf{u})}(\cdot)$, $\boldsymbol{\mu}^{(v)}(\cdot)$ and $\boldsymbol{\tau}^{(v)}(\cdot)$ are the mean and variance representations of the direction and length variables respectively, $\mathcal{N}(\cdot)$ is a Gaussian distribution and $\mathcal{N}_s(\cdot)$ is a distribution defined on a D-sphere. Samples from both distributions can be transformed into VFs with the inverse transformation to that used above, $\nabla\mathbf{h}(t) = \exp\{v(t)\}\mathbf{u}(t)$.

In principle one may wish to constrain the variance of Stochastic VF (SVF) to ensure that $\mathbf{u}(t) \cdot \boldsymbol{\mu}^{(\mathbf{u})}(t) > 0$ since, with large variance, samples from the directional distribution (eq. (6)) will be reversed with respect to the expected direction of the VF. We believe this directional preservation property is important since it preserves the expected inductive bias of the SVF for each sample, which tends to give lower variance samples. A similar motivation led to us imposing uncertainty on length in the log domain. Although noise can be imposed on VFs without the decomposition (*e.g.* see Otto et al. [11]), the proposed decomposition naturally orientates the distribution along the expected direction of the VF.

2.3 Loss functions

Two factors directly affect the Number of Function Evaluations (NFE) when solving IVPs: the total distance moved by the particle and the 'straightness' of its trajectory. In this section we motivate several losses, and in particular transportation and variance losses have a minimising effect NFEs.

Mixture Density Loss (MDLoss). SVFM models produce mixture distribution as outputs, and these are reconciled into a mixture density loss as follows

$$\mathcal{L}_{\text{MDLoss}} = -\log \sum_{k=1}^{K} p\left(\mathbf{h}\left(t_i\right) | \boldsymbol{\mu}^{(k)}\left(t_i\right), \boldsymbol{\tau}^{(k)}\left(t_i\right)\right) \boldsymbol{\pi}^{(k)}\left(t_i\right) \tag{8}$$

where p is the density of the output distribution. This loss resembles those given in Gaussian mixture models and mixture density networks [12].

Transportation Loss (TLoss). A regulariser based on path length is defined as follows

$$\mathcal{L}_{\text{TLoss}} = \frac{1}{T} \sum_{i=1}^{T} \|\mathbf{h}(t_i) - \mathbf{h}(t_{i-1})\|_2^2 \tag{9}$$

The minimal possible $\mathcal{L}_{\text{TLoss}}$ between $\mathbf{h}(t_0)$ and $\mathbf{h}(t_T)$ is $\|\mathbf{h}(t_T) - \mathbf{h}(t_0)\|_2^2$ and this is achieved when all VF evaluations point in the same direction, *i.e.* $\nabla\mathbf{h}(t_i) = \nabla\mathbf{h}(t_0)s_i \; \forall s_i \in \mathbb{R}_+$.

Variance Loss (VLoss). VFs can be encouraged to point in the same direction by regularising their variance. Letting $\mathbb{E}[\nabla\mathbf{h}(t)] = \frac{1}{T}\sum_{i=1}^{T} \nabla\mathbf{h}(t_i)$ be the expected direction, VF variance loss is

$$\mathcal{L}_{\text{VLoss}} = \frac{1}{T} \sum_{t=1}^{T} \|\nabla\mathbf{h}(t_i) - \mathbb{E}\left[\nabla\mathbf{h}(t)\right]\|_2^2 \tag{10}$$

Combined transportation distance and directional variance losses are termed Transport & Variance Loss (TVLoss).

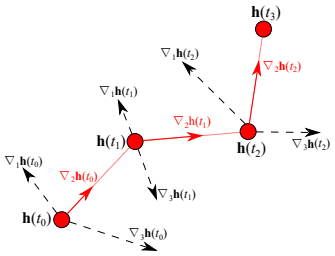
Forecasting Loss (FLoss). In forecasting problems, a datapoint is a sequence of T' measurements and timestamps. We define $\mathbf{X} \in \mathbb{R}^{T' \times D}$ to be the matrix of measurements and $\mathbf{t}_{\mathbf{X}} \in \mathbb{R}^{T'}$ to be the corresponding timestamps. The total path loss is then defined as follows

$$\mathcal{L}_{\text{FLoss}} = \frac{1}{T} \sum_{i=1}^{T} \ell\left(\mathbf{h}(t_i), \text{interp}\left(\mathbf{X}, \mathbf{t}_{\mathbf{X}}, t_i\right)\right) \tag{11}$$

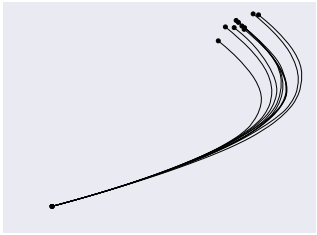
where interp is an interpolation function that estimates data at t_i, and ℓ is a secondary loss that penalises $\mathbf{h}(t_i)$ that are far from the interpolated target. We choose cubic interpolation in our experiments and the choice of ℓ dependent on the model; mean squared error is used with baseline methods and MDLoss is used with the proposed model.

3 Sampling and SVFM relationships

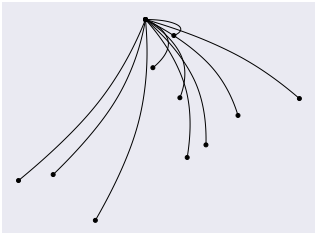
Figure 4(a) illustrates branching patterns that are characteristic of VFM solutions. The initial state $\mathbf{h}(t_0)$ is shown by the red node on the bottom left, and its trajectory to $\mathbf{h}(t_3)$ is highlighted in red. At each evaluation position one of three VFs is dynamically sampled from the distribution $\boldsymbol{\pi}(t_i)$. Although we only consider myopic approaches here, *i.e.* steps are taken based on samples from $\boldsymbol{\pi}(t_i)$, other approaches (including particle filtering and reinforcement learning) will be explored in future work. Exhaustive search is prohibitive since it is exponential on the number of evaluation points. Ten samples from uncertain VFs are shown in figs. 4(b) and 4(c). Scaled VFs (fig. 4(b)) which always initially follow the same trajectory will tend to 'fan out' as curvature is encountered whereas directional uncertainty (fig. 4(c)) tends to fan out immediately. The full factor graph of the proposed approach is shown in fig. 5. NODE units are depicted by the black squares in this figure.



(a) Mixture uncertainty



(b) Length uncertainty



(c) Direction uncertainty

Figure 4: Illustration of three forms of uncertainty on randomly initialised VF units.

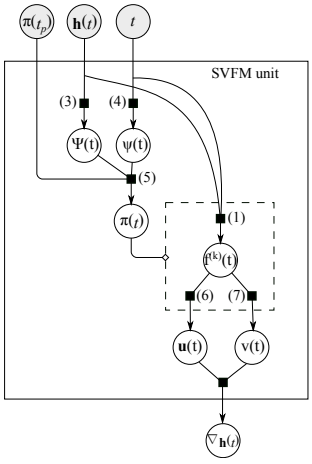


Figure 5: SVFM unit.

The proposed model, SVFM, encapsulates all of the properties shown in fig. 5. Sequences of samples from SVFMs are likely to vary significantly even under the same contextual conditions, and this will cause problems for ODE solvers. Since the classic Runge-Kutta 4 (RK4) method requires four function evaluations in determining the next direction [13] and the Dormand-Prince (DP) method requires six [14] their convergence guarantees do not hold since SVFMs are discontinuous everywhere [15], but with careful scheduling it is possible for these guarantees to hold at each step. Our approach is to record the random state at the beginning of the evaluation campaign, and to reset the random number generator's state to the recorded state whenever samples need to be drawn during the campaign, and these are used with the reparameterisation trick on Gaussian and discrete distributions [16, 17]. Experimentally, this trick significantly reduces the number of function evaluations required with DP solvers that are used in our experiments. The incorporation of uncertainty and mixture distributions to VFs are the key factors that help solve scaling and splitting cases and also facilitate behavioural modelling. Figure 6 highlights the relationship between stochastic (S-, blue), mixture (-M, black) and augmented (A-, red) [6] models. This figure not only shows that these three modelling choices can be used together, but that any combination can be selected.

Although the most complete model is A-SVFM as it incorporates all three components, our experimentation will primarily be evaluated with SVFM models because our interest is in forecasting and augmentation lifts the representation out of the original feature space.

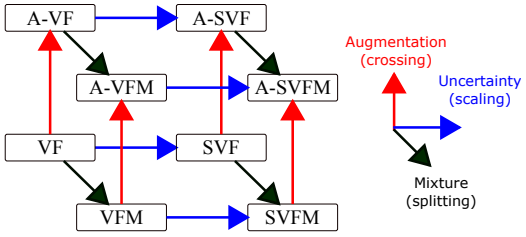


Figure 6: Relationships between discussed models.

We study some properties of these model classes theoretically. First we prove that NODE and ANODE models are unable to model splitting and scaling problems. We belive this further justifies our work since these problems are basic and (conceptually) simple components. Secondly we characterise the nature of VFs that require minimal NFEs. Although it turns out that the resulting VFs are trivial and are not useful in general, the analysis provides justification for the variance and transportation losses mentioned in the previous section. Details of these analyses can be found in the supplementary material.

4 Results and discussion

Illustrative examples. Figure 7 depicts transformation of VF (top), VF with TVLoss (middle; herein called VF-TVLoss) and SVFM (bottom) models on the nested circles dataset. The NODE solution has learnt a complex trajectory that requires the inner circle to 'break' through the outer circle. We note that particles are transported large distances through many NFE to break through, and this jeopardises generalisation accuracy. We can reduce NFE by incorporating TVLoss. Their effect is visibly seen on the top left of the middle figure where several blue points remain stationary at their

original positions and all red trajectories are close to linear. However, in solving this problem many blue datapoints have been transported *around* the outer circle and these nonlinear trajectories require many NFEs. SVFM (bottom) can be seen to produce linear transformations since the trajectory of any two particles are independent if assigned to different VF components.

Although the example in fig. 7 (bottom) clearly demonstrates that SVFM models facilitate linear trajectories that need fewer function evaluations, the learnt model in this particular case has discovered a trivial solution since $\boldsymbol{\pi}(t_0)$ is directly associated to the class label. In the supplement we illustrate a more comprehensive example on the Exclusive-Or (XOR) dataset. SVFM cannot learn a trivial solution of this dataset but will still produces solutions requiring fewer NFEs than baselines. However, it is more challenging to visually assess linear preservation due to the nature of the XOR dataset.

The integration variable t and the interval over which it is integrated are abstract quantities that directly affect the depth of the network. In our application, we interpret t as the Time of Day (ToD) and instead of parameterising our VF networks with the time variable t we use a time tuple $\mathbf{t} = (\cos(t/l), \sin(t/l))$ ($l = \frac{\pi}{12}$ if t is given in hours) since it has smooth and periodic characteristics for increasing t.

We call the augmentation of t to $\mathbf{t}$ cyclic VF continuation and an example of it is illustrated in fig. 8. Here, the function $g(t) = \sin(t) + t/10$ (shown in the solid trace) is learnt by a NODE mode with data from the blue interval ($0 \leq t \leq 2\pi$). The model is then tested by querying the endpoint at $t^* = 10\pi$, a point well outside the training domain. When determining the endpoint, the NODE model performs several function evaluations shown with the blue and red dots. We see that the periodic traits of the function have been extrapolated well beyond the training interval into the red region. Of particular note is that all intermediate datapoint evaluations are faithful to the true function which is directly encouraged by the forecasting loss. In the context of behavioural modelling, this simple example demonstrates that cyclic VF continuation should allow for periodic behavioural traits to be representatively modelled over long forecast horizons. Although the demonstration here is rather straightforward, richer notions of meaningful periodic dependency will be captured from real datasets and tasks. Traditional VF parameterisations cannot learn even this basic extrapolation task.

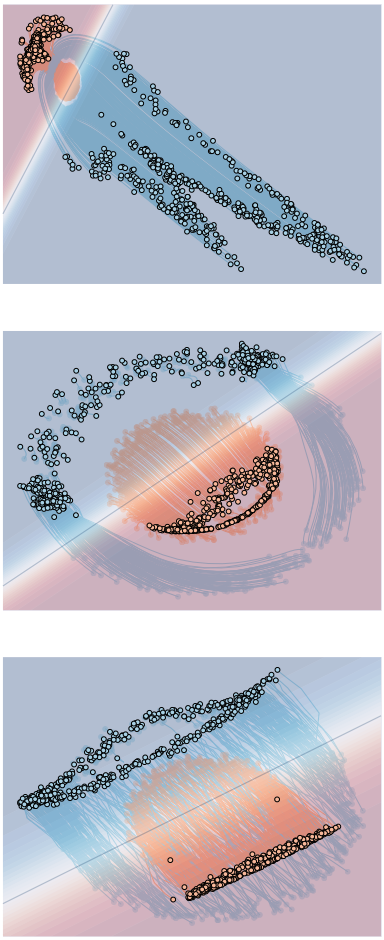


Figure 7: Nested circle predictions.

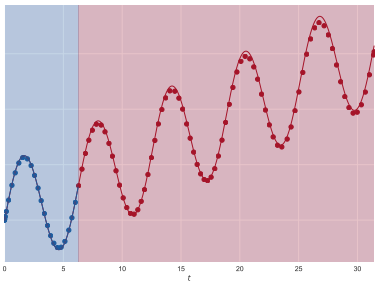
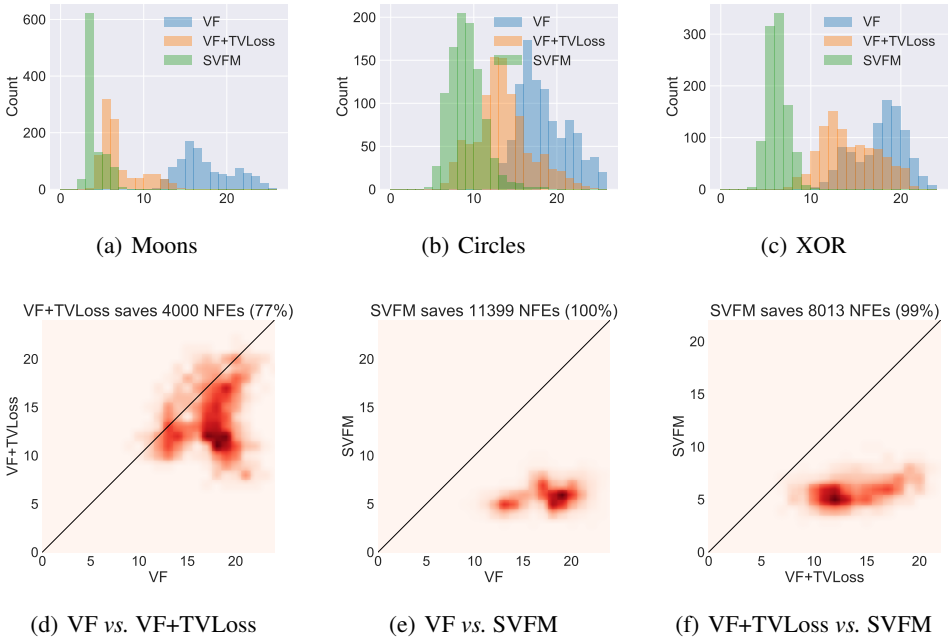


Figure 8: Cyclic continuation.

Forward evaluation analysis. The effective complexity of NODE models is widely reported with the NFE metric, but NFE is a somewhat nebulous term since it more accurately captures the maximal NFE of a batch. We illustrate this by solving the IVP for every instance of a dataset separately and producing a histogram on the set of NFEs. Taking the particular example of the VF model on the moons dataset (*i.e.* blue histogram in fig. 9(a)) the NFE metric (in the traditional sense) is 26, but it is clear that most datapoints require much fewer function evaluations, with the mean and median ≈ 17. Computational savings can be achieved by separating the 'hard' and 'easy' instances from one another since it is the hardest instance of a batch that subjects the rest to unnecessary evaluations. Estimation error given by embedded solvers allow for dynamic evaluation.



(a) Moons (b) Circles (c) XOR

(d) VF *vs.* VF+TVLoss (e) VF *vs.* SVFM (f) VF+TVLoss *vs.* SVFM

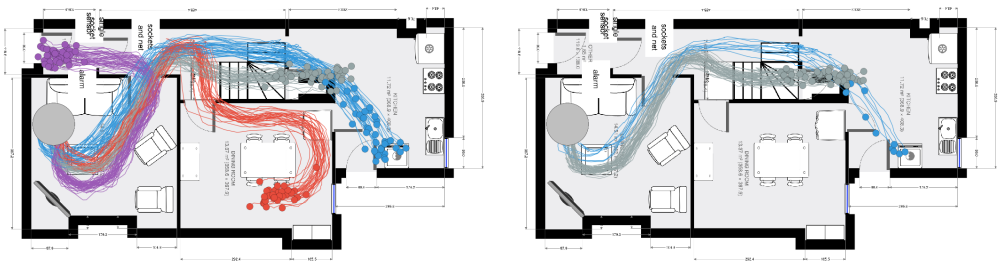
Figure 9: NFE histograms for baseline and proposed models on moons (fig. 9(a)), circles (fig. 9(b)) and XOR (fig. 9(c)) dataset. The per-instance NFE heatmap is shown for the XOR dataset in figs. 9(d) to 9(f). Here, red indicates regions high density and the leading diagonal is the line of equal NFEs. SVFM models consistently require fewer NFEs. The title of these figures tallies the total NFE savings and proportions of instances in receipt of these savings.

The remining figures in the top row of fig. 9 present the NFE histograms on three datasets, and baseline, VF+TVLoss and SVFM results are shown in blue, orange and green respectively. The basic NODE models are shown here to require the most function evaluations. Although incorporating TVLoss reduces the number of samples required, the greatest savings in all cases are obtained with the SVFM model since it directly facilitates linear solutions.

The NFE distributions from pairs of models are compared in the lower row of fig. 9 on a per-instance basis. Here, the (i, j)-th component of the matrix counts the number of instances for which the first model made i evaluations and the second made j. The NFEs for VF, VF+TVLoss and SVFM models are compared on the XOR dataset. This figure shows that NFE savings are gained by using the proposed method and TVLoss losses over baselines, with the biggest savings delivered by SVFM. In particular with figs. 9(e) and 9(f) we see that the proposed method dominates VFs in all cases. The supplement contain per-instance NFE distributions on the other datasets investigated and, in general, show more significant savings with the SVFM approach.

Behavioural forecasting. In behavioural forecasting we are interested in predicting the future location and trajectory of a person given knowledge of their original starting position. SVFMs utility in delivering these forecasts is demonstrated with a bespoke dataset captured in a residential environment. A volunteer was recorded with a Light Detection And Ranging (LiDAR) data collection unit and the resulting pointcloud was processed with Simultaneous Localisation And Mapping (SLAM) techniques to produce relative time and location data. The protocol is detailed in the supplement. For the purposes of this experiment, data was collected over four consistent paths uniformly that start in the living room and end at the front door, kitchen, landing and dining room. The behavioural data will be released on publication in raw and processed forms.

In fig. 10(a) the floor plan of the residential environment and the approximate location of furniture are shown. An SVFM model is learnt on the data above with the forecasting loss (eq. (11)), and 200 paths sampled from the learnt model are overlaid on this figure. Four clear trajectories corresponding to the four endpoints mentioned above can be seen in the figure and are colour coded according to endpoint. Although the essential trajectory of the paths is the same, each particular path is unique.



(a) Forecasts learnt from uniform endpoints. (b) Conterfactual forecast based on time of day.

Figure 10: Visualisation of behavioural forecasting starting from the couch in the living room (grey circle). Purple, blue, grey and red represent front door, kitchen, landing (up stairs) and dining room trajectories respectively.

This is due to directional and length uncertainty. Length uncertainty is particularly notable with the kitchen trajectories (blue) in which endpoints span a region of approximately two metres.

The distribution over endpoints was balanced in our data capture, and hence in fig. 10(a). Although this clearly illustrates the model's capability in forecasting behaviour, it does not represent the dynamics of naturalistic behaviour since people will not enter rooms uniformly at random in general. In order to demonstrate the model's ability to capture naturalistic behaviour, we synthesise a dataset where the encountered paths are conditioned on the ToD. 'Day' and 'night' periods are defined, and during the daytime all paths are walked with uniform probability. During 'night,' however, the landing and kitchen are selected with probability 0.9 and 0.1 respectively. An SVFM model influenced by the time tuple is learnt on these paths. We test with the same conditions of fig. 10(a) during the 'day,' and endpoints mimic those in fig. 10(a). However, a counterfactual query of the following form was made: 'what would the endpoints be had the ToD been night instead of day?' The resulting paths are shown in fig. 10(b), and the model has delivered on the expectation that 'landing' endpoints should be favoured given the temporal context of the query. This is a more complete demonstration of the cyclic continuation demonstrated earlier in fig. 8.

Assessment of forecasting models like this are of particular interest in healthcare domains. A growing pressure for passive assessment disease state and prognosis is leading to better characterisation of symptom such as 'wandering behaviour' for patients with Alzheimer's disease. This symptom is characterised by high-entropy transit within the home, and this is exactly what is shown in fig. 10(a). These are easily distinguished from the low-entropy transitions such as those shown in fig. 10(b). Baseline NODE and ANODE models are unable to adequately characterise these behavioural problems due to the natural variation of walked paths in these domains.

5 Conclusion

This paper is concerned with encouraging parsimonious advections from neural ordinary differential equation models, and we draw two main conclusions in this work that are fortified by our experiments: 1. loss functions acting on particle trajectories lead to models with lower effective complexity; and 2. introducing uncertainty to vector fields directly delivers simpler solutions. To the best of our knowledge, trajectory-based losses have not previously been explored within the NODE paradigm, and the three that we introduce are shown here to be a straightforward and effective means of controlling model complexity. The most successful model in our experiments is the stochastic vector field mixture that we propose, and it is the vector field mixtures in particular that facilitate improved performance and linearly transported solutions. The utility of these models is demonstrated on illustrative tasks and in human behavioural modelling where we demonstrate its ability to capture the rich variation that is characteristic and forecast human movement. These results in particular suggest that these models may be applied to open behavioural quantification questions in healthcare settings, particularly with Alzheimer's disease. Future work will build on what is presented here, and in particular will characterise between-resident effects in forecasting and explore behavioural modelling when location is uncertain or latent.

Acknowledgments

This research was conducted under the 'Continuous Behavioural Biomarkers of Cognitive Impairment' project funded by the UK Medical Research Council Momentum Awards under Grant MC/PC/16029.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*, 2017.

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

[4] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272*, 2018.

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6572–6583, 2018.

[6] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented Neural ODEs. *arXiv preprint arXiv:1904.01681*, 2019.

[7] Tom Minka and John Winn. Gates. In *Advances in Neural Information Processing Systems*, pages 1073–1080, 2009.

[8] Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[9] Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.

[10] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.

[11] Mathias Otto, Tobias Germer, Hans-Christian Hege, and Holger Theisel. Uncertain 2d vector field topology. In *Computer Graphics Forum*, volume 29, pages 347–356. Wiley Online Library, 2010.

[12] Christopher M Bishop. Mixture density networks. Technical report, Citeseer, 1994.

[13] Wilhelm Kutta. Beitrag zur näherungweisen integration totaler differentialgleichungen. 1901.

[14] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

[15] Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*, volume 10. Cambridge University Press, 2019.

[16] Francisco R Ruiz, Michalis Titsias RC AUEB, and David Blei. The generalized reparameterization gradient. In *Advances in neural information processing systems*, pages 460–468, 2016.

[17] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[18] Ralph Howard. The gronwall inequality. *lecture notes*, 1998.

[19] John Charles Butcher and Nicolette Goodwin. *Numerical methods for ordinary differential equations*, volume 2. Wiley Online Library, 2008.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1271–1278. IEEE, 2016.

[22] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 978-0-262-20162-9.

Appendices

A Theoretical analysis

A.1 NODE and ANODE fail cases

In this section we introduce some theoretical discussions on ODEs and NODEs in solving crossing, splitting and scaling problems. We draw from existing work that shows NODEs cannot solve crossing and also demonstrate that neither NODEs nor ANODEs can solve splitting and scaling.

The following notation is used: $\mathbf{h}(t_T) \in \mathbb{R}^D$ represents the solution of the IVP problem with vector fields defined as $\nabla \mathbf{h}(t) = f(\mathbf{h}(t), t; \boldsymbol{\theta}) \in \mathbb{R}^D$ over the integration interval $[0, t_T]$. $\mathbf{h}_i(t)$ indicates the transformation at t of a particular instance, i. We set up by established standard conventions by omitting the parameterisation of $\boldsymbol{\theta}$ on f for notational convenience and f is assumed to be Lipschitz continuous, *i.e.* $\forall t$ there exists a positive C such that

$$\|f(\mathbf{h}_1(t), t) - f(\mathbf{h}_2(t), t)\| \leq C\|\mathbf{h}_1(t) - \mathbf{h}_2(t)\|. \tag{12}$$

Proposition. *Let* $\mathbf{h}_1(T)$ *and* $\mathbf{h}_2(T)$ *be two solutions of an ODE with different initial conditions,* i.e. $\mathbf{h}_1(0) \neq \mathbf{h}_2(0)$. *Then, for all* $t \in (0, T]$, $\mathbf{h}_1(t) \neq \mathbf{h}_2(t)$.

Proof. Proof follows naturally by contradiction based on the existence and uniqueness of ODE solutions. If there exists some $t^* \in (0, T]$ for which $\mathbf{h}_1(t^*) = \mathbf{h}_2(t^*)$, then let $\mathbf{h}_3(t^*)$ be a new IVP defined at t^*. Since existence and uniqueness of ODEs hold when solving IVPs forward and backward, the backward solution $\mathbf{h}_3(0)$ is unique. This contradicts the setup where $\mathbf{h}_1(0)$ and $\mathbf{h}_2(0)$ are distinct, and thus the proposition is true.

This proposition is useful in this setting since it demonstrates that ODE solutions can never intersect because otherwise the fundamental existence and uniqueness properties of ODEs are violated. ODEs, therefore, have no solutions for crossing, splitting and scaling problems.

This theory is generalised to NODEs by Dupont et al. [6]. We do not duplicate the proof here, but assuming that f is Lipschitz continuous they show that the feature mappings are homeomorphisms and makes use of Gronwall's lemma [18] in the proof. With this established, it is trivial to generalise corollaries that demonstrate that crossing, splitting and scaling problems cannot be solved by NODEs.

Augmented NODEs were introduced to solve the crossing problem by concatenating the data with zeros and solving the IVP with this new augmented representation. ANODEs flows may use the additional dimension(s) in finding solution paths which provide a solution for the crossing problem. Let us formally analyse NODE's ability to tackle splitting and scaling. Let $g_s : \mathbb{R} \to \mathbb{R}$ be a function such that

$$g_s = \begin{cases} \text{-1 with probability 0.5} \\ \text{1 with probability 0.5} \end{cases} \tag{13}$$

Proposition. *ANODE flows cannot model* g_s.

Proof. The proof of this follows directly from the fact that ANODEs retain the theoretical properties of NODEs and thus their feature mappings are homeomorphisms [6]. Therefore, they do not solve g_s since it is a one-to-many mapping.

A proof for the scaling function follows naturally with the same rationale. The key problems is that since the initial conditions of the data are identical, their paths will remain the same (even with augmentation) and will never become untangled. Our proposed model overcomes this limitation by assigning data to a vector field probabilistically which allows eq. (13) to be modelled.

A.2 VF variance and alignment

Aligned VFs are encouraged by the variance reduction loss discussed in the main text, and in this section we expand on the computational gains achieved when solving IVPs with aligned VFs.

We formalise this analysis with Runge-Kutta-based iterative embedded ODE solvers. These can be characterised by their Butcher tableau [19] that defines the coefficients, temporal offsets and evaluation weighting of VFs. The Butcher tableau of an order s embedded solver is often seen visually as

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \\ & b_1^* & b_2^* & \dots & b_s^* \end{array}$$

and for a given state, $\mathbf{h}(t_n)$, the next output is given by

$$\mathbf{h}(t_{n+1}) = \mathbf{h}(t_n) + h \sum_{i=1}^{s} b_i \mathbf{k}_i \tag{14}$$

where the intermediate vector field evaluations are defined as

$$\mathbf{k}_i = f\left(\mathbf{h}(t_n) + \sum_{j=1}^{s} a_{ij}\mathbf{k}_j, t_n + c_i h\right) \tag{15}$$

and a, b and c are as defined in the tableau above and h is the step size for that iteration. The final row of the tableau is used to estimate truncation error as follows

$$\mathbf{h}^*(t_{n+1}) = \mathbf{h}(t_n) + h \sum_{i=1}^{s} b_i^* \mathbf{k}_i \tag{16}$$

$$\begin{aligned} e_{t_{n+1}} &= \mathbf{h}(t_{n+1}) - \mathbf{h}^*(t_{n+1}) \\ &= h \sum_{i=1}^{s} (b_i - b_i^*)\mathbf{k}_i \end{aligned} \tag{17}$$

and the step size h is adjusted dynamically when deriving the solution to be close to a specified tolerance value. If the error is too high, h is reduced and the $(n+1)$-th iteration is repeated. If the error below the tolerance, $\mathbf{h}(t_{n+1})$ is accepted but the step size may be increased if the error is too small.

If the modelling framework successfully aligns all VFs then the error term becomes

$$e(t_{n+1}) = \mathbf{k}_* \, h \sum_{i=1}^{s} (b_i - b_i^*) \tag{18}$$

where $\mathbf{k}_*$ is the aligned direction and the truncation error has become independent of the intermediate evaluation points. By design $\sum_i b_i = 1$ and $\sum_i b_i^* = 1$ and so the truncation error tends to 0 in the equation above. This allows solutions to be discovered with just one step. Although the systems characterised by eq. (18) are not particularly interesting in general, we introduce it since it is minimal in the sense of NFE requirements. In our experimentation with variance reduction we observe significant savings using these losses in our evaluation, which can be seen to tend towards this limiting case. Interestingly, this suggests a link between transportation and variance losses since under the conditions of eq. (18) both are minimal.

B Experimental details

B.1 Models

Several architectural and optimisation parameters must be specified in all experiments. These are listed here:

- VF representer
  - Number of hidden layers: 1 and 2.
  - Number of hidden units: 32 and 64 (originally 16 was considered, but it was omitted since it was never selected).
  - Activation function: rectified units in all cases.
  - Output activation: linear units.
- Optimisation
  - Optimizer: Adam [20].
  - Batch size: 50 and 100.
  - Learning rate: 10^{-2} and 10^{-3}.
- ODE solver
  - Tolerance: 10^{-6}.
- SVFM
  - Component selection: pick and stick, forward filtering.

We introduced four loss functions in the main paper: Mixture Density Loss (MDLoss), Transportation Loss (TLoss), VLoss and Forecasting Loss (FLoss). Since FLoss penalises deviation from a path, it is incompatible with TLoss and VLoss. Classification tasks, however, may mix TLoss and VLoss losses with predictive losses (*e.g.* cross-entropy). The balance between the predictive loss and the path regularisation is established with one parameter λ, and this is selected on the validation set.

B.2 Data

B.2.1 Synthetic data

Three example datasets are used in this paper: moons, nested circles and XOR. In all cases 1,000 datapoints are sampled for training, testing and validation. The top row of fig. 12 illustrates their configuration with the positive (red) and negative (blue) classes.

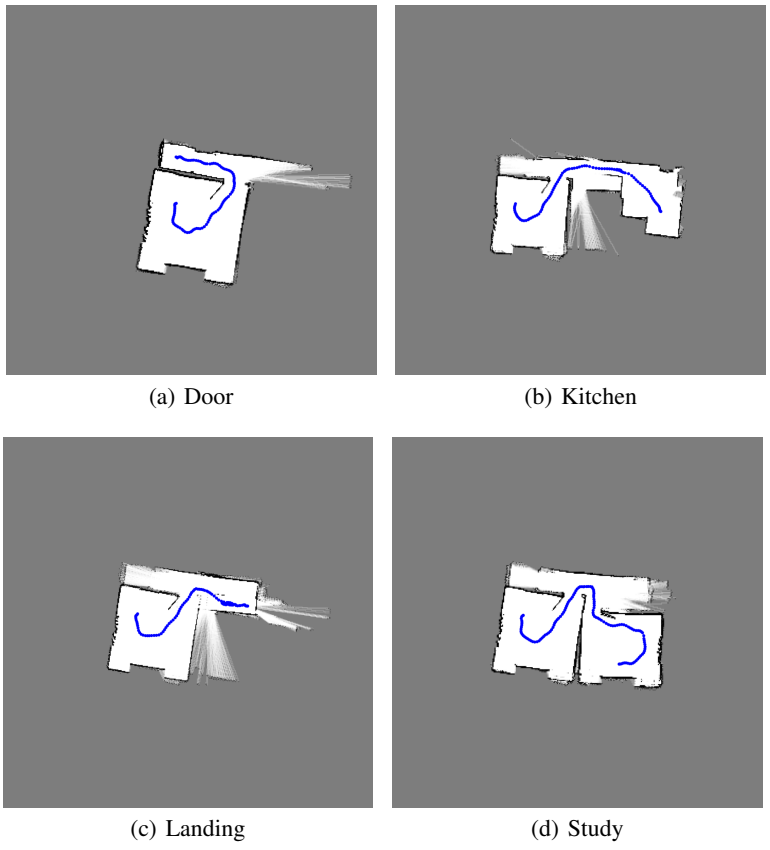
B.2.2 Behavioural data collection and processing

The behavioural dataset was collected in an adapted residential house. The collection mechanism involved a bespoke robotic wearable, capable of gathering 2-dimensional LiDAR point clouds from a human participant. The experimental procedure asked the participant to walk naturally into 4 different sections of the house from the same origin point, located in the living room. The 4 different targets in the house included the front door, the study room, the kitchen, and the landing of the stairs. Time of each experiment was assured through NTP synchronisation between the wearable and university NTP servers.

The procedure for collection followed a protocol:

1. Begin in the origin 'anchor'.
2. Choose one of the four target locations.
3. Proceed to the target. This should take roughly 5-10 seconds.
4. Repeat 10 times.

The subsequent processing of the point cloud was performed using 2-dimensional SLAM. Using MATLAB software, the data was associated using a loop closure method outlined in [21]. The map and the locations were extracted using an occupancy grid map [22]. The loop closure in this context



(a) Door (b) Kitchen

(c) Landing (d) Study

Figure 11: Visualisation of SLAM maps mapping a single trajectory from the living room to the four endpoints considered in this work. We can see that the map derived from each setting is a faithful representation of the floorplan shown in the main paper.

specifies a method of location error minimisation from aggregated sub maps [21]. The parameters of the algorithm included laser reflectivity, internal loop closure threshold, resolution of the grid map and down-sampling of the data.

Examples of the produced maps, for all 4 targets can be seen in fig. 11. The locations extracted using SLAM were required to be additionally associated together using Iterative Closest Point (ICP). This was done to make sure that the extracted walking tracks overlapped. The extracted location information further served as the input to the model. Some of the variation at the beginning of the paths in figs. 10 (a) and 10(b) are due to the measurement uncertainty from the low-cost LiDAR and SLAM processing.

The raw and processed data will be made available on a public repository after publication.

C Supplementary results

The figures in this section (figs. 12 and 13) are included to support and provide deeper intuition of the results given in the main paper. The captions provide self-contained descriptions of the main discussion points and conclusions that can be drawn from the pictures.

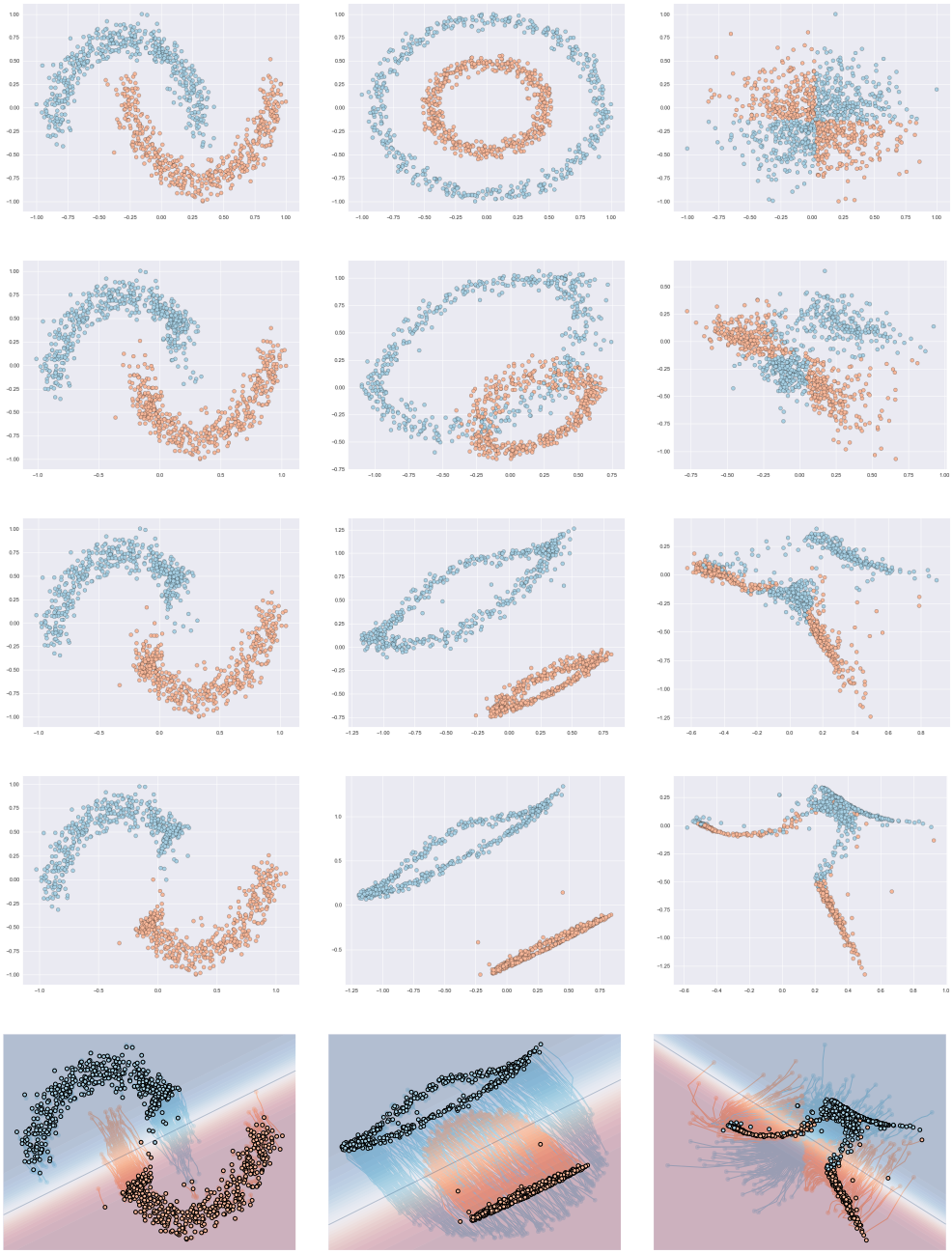
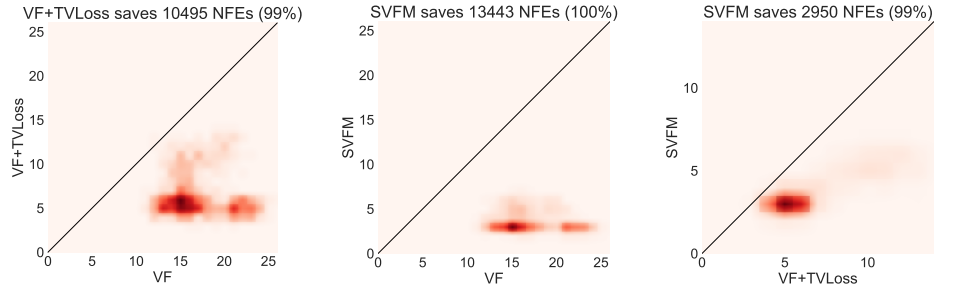
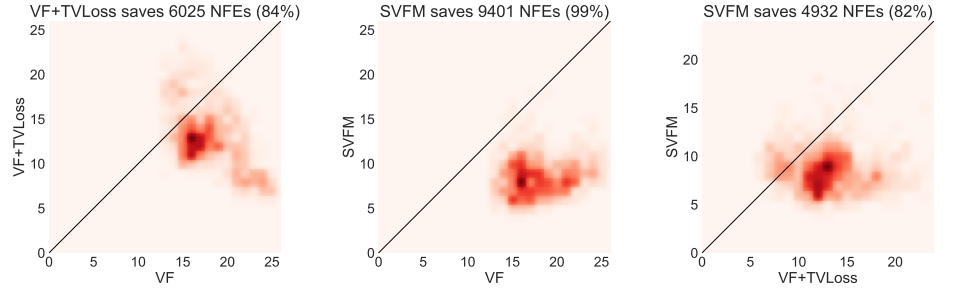


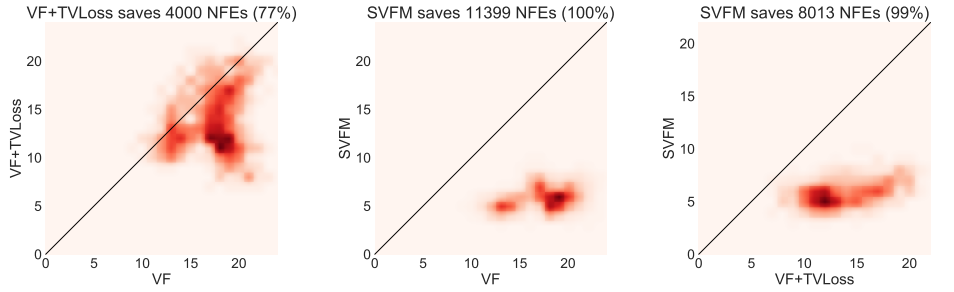
Figure 12: Visualisation of SVFM solutions on the moons (left column), nested circle (middle column) and XOR (right column). Each row depicts a point through the solution path, and the bottom row shows the predictive distribution on the final transformation. Notice that the total particle movement of the moons solution is minimal. In the second row of the nested circles dataset the position of two classes overlap but in the next row we see that have split apart. This is because the two classes have been assigned to different vector fields. The vector field mixture helps the XOR dataset by learning to assigning blue quadrant on the top right to one component and the rest to the other. This quadrant is free to move upwards independently from the rest. The remaining clusters are assigned to the same VF and have learnt to split apart so that the lower left quadrant may join the first. Best viewed on a digital device.



(a) Moons VF *vs.* VF+TVL (b) Moons VF *vs.* SVFM (c) Moons VF+TVL *vs.* SVFM



(d) Circles VF *vs.* VF+TVL (e) Circles VF *vs.* SVFM (f) Circles VF+TVL *vs.* SVFM



(g) XOR VF *vs.* VF+TVL (h) XOR VF *vs.* SVFM (i) XOR VF+TVL *vs.* SVFM

Figure 13: Per-instance NFE comparison densities on all illustrative experiments. The top, middle and lower rows arise from the moons, nested circles and XOR datasets respectively, and the left, middle and right columns compare VF *vs.* VF+TFLoss, VF *vs.* SVFM and VF+TFLoss *vs.* SVFM. In all cases significant savings are achieved with TFLosses and with SVFM models, with savings of $\approx$ 3,000-13,000 function evaluations over baseline approaches (or savings of 3-13 function evaluations on every instance, on average).