Turning Normalizing Flows into Monge Maps with Geodesic Gaussian Preserving Flows

Guillaume Morel
IMT Atlantique
guillaume.morel @ imt-atlantique.fr

Lucas Drumetz
IMT Atlantique
lucas.drumetz @ imt-atlantique.fr

Simon Benaichouch
IMT Atlantique
simon.benaichouch @ imt-atlantique.fr

Nicolas Courty
IRISA
nicolas.courty @ irisa.fr

François Rousseau
IMT Atlantique
francois.rousseau @ imt-atlantique.fr

Abstract

Normalizing Flows (NF) are powerful likelihood-based generative models that are able to trade off between expressivity and tractability to model complex densities. A now well established research avenue leverages optimal transport (OT) and looks for Monge maps, i.e. models with minimal effort between the source and target distributions. This paper introduces a method based on Brenier's polar factorization theorem to transform any trained NF into a more OT-efficient version without changing the final density. We do so by learning a rearrangement of the source (Gaussian) distribution that minimizes the OT cost between the source and the final density. We further constrain the path leading to the estimated Monge map to lie on a geodesic in the space of volume-preserving diffeomorphisms thanks to Euler's equations. The proposed method leads to smooth flows with reduced OT cost for several existing models without affecting the model performance.

1 Introduction

To keep the length of this extended abstract short we have removed some proofs. All the proofs are given in the full version of the paper available here `https://arxiv.org/abs/2209.10873`.

Normalizing flows. A popular class of generative models is Normalizing flows (NF). NF models transform a known probability distribution (Gaussian in most cases) into a complex one allowing for efficient sampling and density estimation. To do so they use a smooth diffeomorphism $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ which maps a target probability distribution μ to the known source distribution $\nu = \mathbf{f}_\# \mu$ [1, 2]. In practice the flow must satisfy the change of variables formula $\log p_\mu(\mathbf{x}) = \log p_\nu(\mathbf{f}(\mathbf{x})) + \log |\det \nabla \mathbf{f}(\mathbf{x})|$. There are many possible parameterizations of $\mathbf{f}$, usually relying on automatic differentiation to train their parameters via first order optimization algorithms. For density estimation applications, training is done by maximizing the likelihood of the observed data [1, 3, 2, 4].

Optimal transport. The question of choosing the "best" transformation among all existing ones, independently from how accurately μ models the target distribution is an important one. One way to make the architecture unique (under appropriate conditions on the two distributions) is to use optimal transport [5, 6, 7, 8], that is to choose the one giving the Wasserstein distance between μ and ν, with a squared $\mathcal{L}_2$ ground cost $W_2^2(\mu, \nu) = \min_{\mathbf{f}} \int_{\mathbb{R}^d} |\mathbf{f}(\mathbf{x}) - \mathbf{x}|^2 d\mu(\mathbf{x}), \quad \nu = \mathbf{f}_\# \mu$. An optimal model minimizes the total mass displacement which can be a desirable property even if it is often a difficult task. In particular Brenier's theorem [9] states that the optimal function $\mathbf{f}$ is the gradient of a scalar convex function, which is widely used in practice.

1.1 Main contributions

Polar factorization. An overlooked implication of Brenier's theorem is the so-called polar factorization theorem, that states that the optimal transport map $\nabla\psi$ can be factorized into the composition of two functions $\nabla\psi = \mathbf{s} \circ \mathbf{f}$, the function $\mathbf{f}$ being some arbitrary smooth map from μ to ν and $\mathbf{s}$ an associated measure preserving function of ν [9]. The idea we exploit is the possibility, from a given flow $\mathbf{f}$ (and its corresponding inverse $\mathbf{g} = \mathbf{f}^{-1}$), to rearrange the distribution ν using $\mathbf{s}$ to obtain a new map reducing the OT cost without changing the distribution given by the push-forward $\mu := \mathbf{g}_{\#}\nu = (\mathbf{g} \circ \mathbf{s}^{-1})_{\#}\nu$. The OT-improved map can then be obtained with the composition $\mathbf{g} \circ \mathbf{s}^{-1}$. Interestingly this property is not as popular as the previous one and to our knowledge is not used when dealing with OT and NF models. Yet normalizing flows can take advantage of this formulation mostly because the distribution ν is known and simple which make it possible to construct architectures preserving ν. We consider the most common case where the known distribution is a standard normal and call such rearranging maps Gaussian Preserving (GP) flows. An important point is that by construction our GP map will only change the OT cost of the model. The target density and therefore the training loss given by the model will stay the same. his allows us to take any pre-trained model and compute the associated Monge map, thus improving the model in terms of OT displacement from the source to the target distribution, without changing the modeled density see Figure 1.

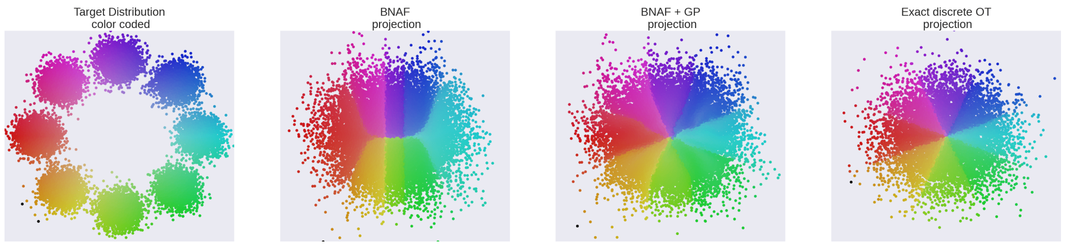

Figure 1: Eight gaussians test case with colored distributions. A GP flow is trained on a pre-trained BNAF model [10] to reduce the OT cost.

Euler's equations. Since several GP flow models can solve the same OT problem, we also look for a way to find the "best" GP flow. In this work, we show that the geodesics associated with the OT problem are actually given by solutions to the Euler equations, following a celebrated result by Arnold [11]. The penalization of Euler's equations in high dimensions and its practical implementation is therefore also considered.

Disentanglement preservation with optimal transport. Finally we show one potential interest of GP flows by studying the preservation of the data structure experimentally. More specifically we focus on the preservation of disentanglement on the dSprites dataset [12] in some variational auto-encoder (VAE) latent space. On this particular example we show that OT allows to preserve the structure of the latent data points which is otherwise destroyed when applying the NF model.

2 Gaussian preserving flows

Polar factorization theorem. The main idea is to use the Brenier's polar factorization theorem to construct the Monge map with a rearrangement of the known probability distribution ν. To preserve the conventions from Brenier's paper [9], we study the OT problem defined from the known probability distribution ν to μ and therefore consider the function $\mathbf{g} := \mathbf{f}^{-1}$.

Theorem 1 (Brenier's polar factorization [9])**.** *Let $(\mathcal{X}, \nu)$ be a probability space, $\mathcal{X} \subset \mathbb{R}^d$ open bounded. Then for each non-degenerate $\mathbf{g} \in L^p(\mathcal{X}, \nu, \mathbb{R}^d)$, there exists a unique convex function $\psi : \mathcal{X} \to \mathbb{R}$ and a measure preserving function $\mathbf{s} : \mathcal{X} \to \mathcal{X}$ such that $\mathbf{g}(\mathbf{x}) = \nabla\psi(\mathbf{s}(\mathbf{x}))$, and $\mathbf{s}(\mathbf{x})$ minimizes the cost $\int_{\mathcal{X}} |\mathbf{g}(\mathbf{x}) - \mathbf{s}(\mathbf{x})|^2 d\nu(\mathbf{x})$.*

Our goal is to leverage the polar factorization theorem in order to solve the OT problem between ν and $\mu := \mathbf{g}_{\#}\nu$ where $\mathbf{g}$ is given and $\nu = \mathcal{N}(\mathbf{0}, \mathrm{Id})$, by looking for the rearrangement $\mathbf{s}$ via an optimization problem. To do so we need to construct a class of measure preserving maps.

Gaussian preserving flows. Consider two probability measures α and β with density h_α and h_β respectively. A map $\mathbf{s}$ is measure preserving between α and β if it satisfies the change of variable equality $h_\beta(\mathbf{x}) = h_\alpha(\mathbf{s}(\mathbf{x}))|\det(\nabla\mathbf{s}(\mathbf{x}))|$. In our case, we want $\mathbf{s}$ to be Gaussian preserving therefore $h_\alpha = h_\beta = e^{-\|\mathbf{x}\|^2/2}$ and one gets

$$|\det\nabla\mathbf{s}(\mathbf{x})| = e^{(\|\mathbf{s}(\mathbf{x})\|^2 - \|\mathbf{x}\|^2)/2}. \tag{1}$$

It turns out that Lebesgue preserving functions (i.e. satisfying $|\det\nabla\boldsymbol{\phi}| = 1$) can be used to construct maps satisfying (1). In the following we will denote $\mathbf{erf} : \mathbb{R}^d \to \mathbb{R}^d$ the distribution function of a one dimensional Gaussian (that is $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{t^2}dt$) applied component wise.

Proposition 1. *Let $\mathbf{s}$ be a smooth preserving function (i.e. satisfying (1)). Then there exists $\boldsymbol{\phi} : (-1,1)^d \to (-1,1)^d$ such that $|\det\nabla\boldsymbol{\phi}| = 1$ and $\mathbf{s}(\mathbf{x}) = \sqrt{2}\,\mathbf{erf}^{-1}\circ\boldsymbol{\phi}\circ\mathbf{erf}(\frac{\mathbf{x}}{\sqrt{2}}), \quad \mathbf{x}\in\mathbb{R}^d$.*

From now on we will focus on the construction of volume and orientation preserving maps (i.e. satisfying $\det\nabla\boldsymbol{\phi} = 1$) since functions satisfying $\det\nabla\boldsymbol{\phi} = -1$ can be constructed from them. It is also possible to show that under simple hypothesis on the Monge map and the NF architecture, the GP flow $\mathbf{s}$ is C^1 and either $\det\nabla\boldsymbol{\phi} = 1$ everywhere or $\det\nabla\boldsymbol{\phi} = -1$ everywhere.

2.1 Volume-orientation preserving maps

First we introduce the space $\mathrm{SDiff}(\Omega)$ we will working with from now on. Let $\mathrm{Diff}(\Omega)$ be the set of all diffeomorphisms in Ω then $\mathrm{SDiff}(\Omega) := \{\boldsymbol{\psi}\in\mathrm{Diff}(\Omega),\ \det(\nabla\boldsymbol{\psi})(\mathbf{x}) = 1,\ \forall\mathbf{x}\in\Omega\}$, where $\Omega = (-1,1)^d$. That is we need a transformation which satisfies two properties: 1) the function must be volume and orientation preserving, 2) the solution must stay in the domain $(-1,1)^d$. Consider the following ODE:

$$\begin{cases}\frac{d}{dt}\mathbf{X}(t,\mathbf{x}) = \mathbf{v}(t,\mathbf{X}(t,\mathbf{x})), & \mathbf{x}\in\Omega, \quad 0\le t\le T,\\ \mathbf{X}(0,\mathbf{x}) = \mathbf{x}.\end{cases} \tag{2}$$

We impose two conditions on the velocity $\mathbf{v}$:

$$\nabla\cdot\mathbf{v} = 0, \quad \text{in } \Omega, \tag{3}$$

$$\mathbf{v}\cdot\mathbf{n} = 0, \quad \text{on } \partial\Omega, \tag{4}$$

where $\mathbf{n}$ is the outward normal at the boundary of Ω. We define $\boldsymbol{\phi}$ to be the solution at the final time $\boldsymbol{\phi}(\mathbf{x}) := \mathbf{X}(T,\mathbf{x})$. Property (3) implies that $\det\nabla\boldsymbol{\phi} = 1$, and property (4) ensures that $\boldsymbol{\phi}$ does not escape Ω. Any function in $\mathrm{SDiff}(\Omega)$ can be written as a solution to (2) for $d\ge 3$ [13], for $d = 2$ some pathological cases can be constructed [14].

The incompressible property (3) and the boundary conditions (4) can be exactly implemented in the network in any dimension. Note however that in order to get all the incompressible vector fields, we need to construct at least $d(d-1)/2$ arbitrary scalar functions. See Appendix B for the practical construction of divergence free functions in high dimensions.

2.2 Euler's geodesics

GP flows give a way to compute the Monge map for any trained NF architecture. Many transformations can achieve this goal and the question of finding the best flow among all volume preserving transformations need to be considered.

Arnold's theorem. In 1966, Arnold [11] showed that the flow described by Euler's equations coincides with the geodesic flow on the manifold of volume preserving diffeomorphisms. This theoretical result therefore gives the reason why regularizing our flows with Euler's equations is a desirable property. Mainly that Euler's equations take the path with the lowest energy to reach the final configuration. Consider the Euler equations:

$$\begin{cases}\partial_t\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v} = -\nabla p, & t\in[0,T],\ \mathbf{x}\in\Omega,\\ \nabla\cdot\mathbf{v} = 0, & t\in[0,T],\ \mathbf{x}\in\Omega,\\ \mathbf{v}\cdot\mathbf{n} = 0, & t\in[0,T],\ \mathbf{x}\in\partial\Omega,\\ \mathbf{v}(0,\cdot) = \mathbf{v}_0,\end{cases} \tag{5}$$

where $\mathbf{v} := \mathbf{v}(t, \mathbf{x})$ is the velocity field, $p := p(t, \mathbf{x})$ the pressure and $\mathbf{n} := \mathbf{n}(\mathbf{x})$ the outward normal at the boundary of Ω. We introduce the energy $\mathcal{E}(\mathbf{X}) := \int_0^T \int_\Omega \frac{1}{2} |\partial_t \mathbf{X}(t, \mathbf{x})|^2 d\mathbf{x} dt$. Now assume $\boldsymbol{\phi} \in \text{SDiff}(\Omega)$, Arnold's problem's consists in finding the path $\mathbf{X}(t, \cdot)_{t \in [0,T]}$ in $\text{SDiff}(\Omega)$ joining the identity to $\boldsymbol{\phi}$ which minimizes $\mathcal{E}$:

$$\min_{\mathbf{X}(t,\cdot) \in \text{SDiff}(\Omega)} \mathcal{E}(\mathbf{X}), \quad \mathbf{X}(0, \cdot) = \text{Id}, \quad \mathbf{X}(T, \cdot) = \boldsymbol{\phi}(\cdot). \tag{6}$$

In other words (6) is the geodesic in $\text{SDiff}(\Omega)$ between Id and $\boldsymbol{\phi}$.

Theorem 2 (Arnold [11])**.** *Assuming the existence of a solution to Arnold's problem,* $\mathbf{X}$ *is solution to* (6) *if and only if* $\mathbf{v}(t, \mathbf{x}) := \partial_t \mathbf{X}(t, \mathbf{x})$ *satisfies Euler's equations* (5)*.*

In practice, we choose to penalize Euler's equations along the trajectories of our GP flows as detailed in Appendix A. We then have two terms in our loss function: the negative log-likelihood and the term related to Euler's equations.

3 An example of application: improving disentanglement preservation with optimal transport

Disentangle representations allow to encode the data in a latent space where change in one direction result in the change over one generative factor in the data. Recently the construction of variational auto-encoders (VAE) [15] with disentangle latent space has received much attention [16, 17, 18, 19]. Applying a NF architecture to such latent space may be needed for various tasks such as density estimation, generative process or general interpolation. The latter requires to preserve as much of the data structure as possible. We therefore propose to experimentally study disentanglement preservation of NF with and without OT. To do this we apply a NF architecture (FFJORD [20] in our case) to the VAE's latent distribution and consider the addition of a GP flow. For the disentangle interpolation in the NF target (gaussian) space we consider the same directions which are present in the VAE latent space and are aligned with the axis. This may be a naive approach since these directions may change depending on the source and target distribution but it seems enough here to already see an improvement. Our experiments are run with the β-TCVAE architecture [18] and the latent space dimension is 10.

dSprites test case. The dSprites dataset [12] is made of 64×64 images of 2D shapes procedurally generated from 5 ground truth independent latent factors. These factors are shape, scale, rotation, x and y positions of a sprite. Since the factors are known we can compute a quantitative evaluation of disentanglement and we choose here to consider the metric from [21] on the continuous factors (i.e. all the factors except the shape) for the three criteria: disentanglement, completeness and informativeness. Table 1 shows that FFJORD destroy the latent structure and give the worst disentanglement, completeness and informativeness scores. Adding GP flows allow to recover the same disentanglement score as the initial latent space and get values closer both for completeness and informativeness. Note that the disentanglement score is slightly better with FFJORD+GP than the initial one however this is probably only due to some approximation in the metric used here since there is no reason that GP flows improved the disentanglement compared to the initial latent space. On Table 2 the OT costs are compared and as expected GP flows allow to reduce the OT cost without changing the loss. Interestingly GP flows with no additional regularization do not converge completely to the Monge map because particles get out of the domain at some point making it impossible to continue the training process. We conjecture that this may be due to non-smooth trajectories of our GP flows and a regularization is therefore needed. As shown on Table 2 adding Euler regularization fixed this issue and allows to further reduce the OT cost. To illustrate the preservation of disentanglement some interpolations are also presented in Figures 2 and 3 in Appendix D. As we can see on the initial latent space picture, since the dimensions are sorted with respect to their KL divergence only the first dimensions carry information and therefore each of the first 5 lines correspond to a generative factor. On the contrary when interpolating on the last dimensions the image stays the same. This structure is lost when mapping the latent space to a gaussian with the FFJORD architecture. When adding GP flows we recover this structure and the interpolation better match the initial latent one.

Model	**Disent.**	**Compl.**	**Inform.**
Init. latent space	0.58	**0.81**	**0.55**
FFJORD	0.39	0.26	0.62
FFJORD+GP	**0.60**	0.68	0.61
FFJORD+GP+EULER	0.59	0.67	0.59

Table 1: Quantitative evaluation of disentanglement (higher is better), completeness (higher is better) and informativeness (lower is better). Adding GP flows make the scores closer to the initial ones.

Model	Loss	OT cost
FFJORD	-17.52	10.45
FFJORD+GP	-17.52	5.60
FFJORD+GP+EULER	-17.52	**5.26**

Table 2: Loss (negative log-likelihood) and mean OT costs. GP flows reduce the OT cost without changing the loss. Adding Euler regularization allows to further reduced the OT cost.

References

[1] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[2] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[3] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[4] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[5] Brittany Hamfeldt. Optimal transport. *Youtube videos*, 2019.

[6] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Center for Research in Economics and Statistics Working Papers*, (2017-86), 2017.

[7] Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94, 2015.

[8] C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008.

[9] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44(4):375–417, 1991.

[10] Nicola De Cao, Wilker Aziz, and Ivan Titov. Block neural autoregressive flow. In *Uncertainty in artificial intelligence*, pages 1263–1273. PMLR, 2020.

[11] Vladimir Arnold. Sur la géométrie différentielle des groupes de lie de dimension infinie et ses applications à l'hydrodynamique des fluides parfaits. *Annales de l'Institut Fourier*, 16(1):319–361, 1966.

[12] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[13] A.I. Shnirelman. Attainable diffeomorphisms. *Geometric and functional analysis*, 3:279–294, 1993.

[14] A.I. Shnirelman. Generalized fluid flows, their approximation and applications. *Geometric and functional analysis*, 4(5):586–620, 1994.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

[16] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *BLA*, 2016.

[17] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[18] Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

[19] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

[20] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018.

[21] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018.

[22] Claudio Canuto, Alfio Quarteroni, M. Yousuff Hussaini, and Thomas A. Zang. *Spectral Methods*. Springer Berlin Heidelberg, 2007.

[23] Alfio Quarteroni. *Numerical models for differential problems; 1st ed.* MS&A : modeling, simulation and applications. Springer, Milano, 2009.

A Penalization of Euler's equations in high dimensions

Numerical schemes developed to efficiently solve the Euler equations [22, 23] (mainly for fluid mechanics problems, i.e. for dimensions up to 3) scale badly when the dimension increases. In this work, the solution to Euler's equations is interpreted as the geodesic to reach the solution of the OT problem and the dimension can be arbitrary large. Therefore we approach the equation (5) through a penalization procedure which can be carried out in any dimensions. As explained before the second and third equations in (5) are satisfied by construction in the network.

Our remaining goal is to constrain the network to be a smooth solution to $\partial_t \mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\nabla p$. The left hand side can therefore be written as the gradient of a scalar function and we note that if a vector $\mathbf{w}_{t,\mathbf{x}} \in \mathbb{R}^d$ satisfies $\mathbf{w}_{t,\mathbf{x}} = \nabla p(t, \mathbf{x})$, then its Jacobian is symmetric $\nabla \mathbf{w}_{t,\mathbf{x}} = (\nabla \mathbf{w}_{t,\mathbf{x}})^T$. In order to solve the first equation in (5), we propose to penalize the non-symmetric part of the Jacobian for the total derivative of $\mathbf{v}$. Since a Jacobian-vector product can be efficiently evaluated in high dimensions (unlike the calculation of the full Jacobian which is computationally expensive), we do not calculate directly the Jacobian and use instead the following property of symmetric matrices: if M is symmetric then $\mathbf{y}^T M \mathbf{z} - \mathbf{z}^T M \mathbf{y} = 0,\ \forall \mathbf{y}, \mathbf{z} \in \mathbb{R}^d$. The idea is to sample random vectors $\mathbf{y}, \mathbf{z}$ during the training and to penalize this term for the total derivative, that is to minimize:

$$R(\mathbf{x}) := \mathbb{E}_{\mathbf{y},\mathbf{z}} \left[\int_0^T \left(\mathbf{y}^T (\nabla \mathbf{w}_{t,\mathbf{x}}) \mathbf{z} - \mathbf{z}^T (\nabla \mathbf{w}_{t,\mathbf{x}}) \mathbf{y}) \right)^2 dt \right], \quad \mathbf{y}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathrm{Id}), \tag{7}$$

with $\mathbf{w}_{t,\mathbf{x}} = \partial_t \mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v}$. In practice, we do not compute the full time integral in (7) as it would be computationally too expensive but calculate the penalization only at our time steps discretization.

Approximation of the total derivative. To reduce the computational burden, we do not calculate exactly the total derivative $\mathbf{w}_{t,\mathbf{x}}$ but use an approximation of its Lagrangian formulation instead. More precisely, consider the variable $\mathbf{X}(t, \mathbf{x})$ from (2) that is the position of a particle at time t with initial position $\mathbf{x}$. We recall the equality $\frac{D}{Dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) = \partial_t \mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) + (\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) \cdot$

$\nabla)\mathbf{v}(t, \mathbf{X}(t, \mathbf{x}))$ and therefore choose to approximate the right hand side by using a first order Taylor expansion of $D\mathbf{v}/Dt$:

$$\frac{D}{Dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) \approx \frac{\mathbf{v}(t^{n+1}, \mathbf{X}(t^{n+1}, \mathbf{x})) - \mathbf{v}(t^n, \mathbf{X}(t^n, \mathbf{x}))}{\Delta t}, \quad \Delta t := t^{n+1} - t^n. \tag{8}$$

In practice, Δt is set to $2\sqrt{\varepsilon}$ where ε is the machine precision. This approximation can be easily computed since it requires only the evaluation of the velocity at two positions of a particle.

B Construction of incompressible vector fields in high dimensions

We focus on the vector fields satisfying (3) for arbitrary large dimensions. Property (4) can then be incorporated with very little additional work. All the proofs are given in the full version of the paper [1].

Proposition 2. *Consider an arbitrary vector field* $\mathbf{v} : \mathbb{R}^d \to \mathbb{R}^d$. *Then* $\nabla \cdot \mathbf{v} = 0$ *if and only if there exists smooth scalar functions* $\psi^i_j : \mathbb{R}^d \to \mathbb{R}$, *with* $\psi^i_j = -\psi^j_i$ *such that*

$$v_i(\mathbf{x}) = \sum_{j=1}^{d} \partial_{x_j} \psi^i_j(\mathbf{x}), \quad i = 1, ..., d, \tag{9}$$

where $\mathbf{v} = (v_1, ..., v_d)$.

To impose the boundary conditions (4) one can simply multiply each ψ^i_j by $(x_i^2 - 1)(x_j^2 - 1)$.

Lemma 1. *Let* $\Omega = [-1, 1]^d$ *and consider the functions* $\psi^i_j(\mathbf{x}) = (x_i^2 - 1)(x_j^2 - 1)\widetilde{\psi^i_j}(\mathbf{x})$ *where* $\widetilde{\psi^i_j}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ *are arbitrary scalar functions satisfying* $\widetilde{\psi^i_j} = -\widetilde{\psi^j_i}$. *Then the function* $\mathbf{v}$ *defined in Proposition 2 satisfies* $\nabla \cdot \mathbf{v} = 0$ *and* $\mathbf{v} \cdot \mathbf{n} = 0$ *on* $\partial\Omega$.

The goal here is to have a GPU-friendly construction of the incompressible vector fields given in Proposition 2. In the following we consider stationary divergence free functions but the time variable can be added with no additional work simply by considering functions in $\mathbb{R}^{d+1}$ instead of $\mathbb{R}^d$ (the gradients are still taken only on the space variables though).

Notations. Regarding the notations we will use the operator diag for two distinct cases: 1) When $\mathbf{w}$ is a vector $\operatorname{diag}(\mathbf{w})$ denotes the diagonal matrix obtained from the vector $\mathbf{w}$. 2) When W is a matrix $\operatorname{diag}(W)$ denotes the vector obtained from the diagonal of W. The operation $\cdot$ denotes the scalar product between two vectors. Finally we have adopted the convention that when a scalar multiplies a vector it multiplies each of its component.

Practical construction. Let $\mathbf{u}^n : \mathbb{R}^d \to \mathbb{R}^{d-n}$. We construct a divergence free function with the functions ψ^i_j defined as

$$(\psi^i_j)^n = \begin{cases} u^n_{i-n} - u^n_{j-n}, & \text{if } i, j \geq n+1, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

To construct this divergence free function we define the matrix $(\nabla \mathbf{u}(\mathbf{x}))^n \in \mathbb{R}^{d \times d}$ and the vector $\mathbf{1}^n \in \mathbb{R}^d$ as

$$(\nabla \mathbf{u}(\mathbf{x}))^n_{ij} = \begin{cases} \partial_{i-n} u^n_{j-n}, & \text{if } i, j \geq n+1 \\ 0, & \text{otherwise.} \end{cases}, \tag{11}$$

$$\mathbf{1}^n_i = \begin{cases} 1, & \text{if } i \geq n+1 \\ 0, & \text{otherwise.} \end{cases}.$$

Lemma 2. *Let* $n \in \mathbb{N}$, $n \leq d-2$ *and consider the function* $\mathbf{u}^n : \mathbb{R}^d \to \mathbb{R}^{d-n}$. *Then the vector field* $\mathbf{v}^n : \mathbb{R}^d \to \mathbb{R}^d$ *defined as*

$$\mathbf{v}^n(\mathbf{x}) = (\nabla \mathbf{u})^n \, \mathbf{1}^n - [\operatorname{diag}(\nabla \mathbf{u})^n \cdot \mathbf{1}^n]\mathbf{1}^n, \tag{12}$$

is divergence free.

[1] https://arxiv.org/abs/2209.10873

To construct the functions (12) we need 1) to compute the product between the Jacobian of a vector valued function and a constant vector 2) sum the diagonal elements of the Jacobian matrix. Both of these operations can be done efficiently on GPU. Note that in order to satisfy the boundary conditions $\mathbf{v}^n \cdot \mathbf{n} = 0$ one can modify the equation (12) as in Lemma 1 to obtain

$$\begin{aligned} &\mathbf{v}^n(\mathbf{x}) = (\mathbf{x}^2 - 1)\odot \\ &\left[2M^n\mathbf{x} + (\nabla\mathbf{u})^n(\mathbf{x})(\mathbf{x}^2-1) - ((\mathbf{x}^2-1)\cdot \operatorname{diag}(\nabla\mathbf{u}(\mathbf{x}))^n)\mathbf{1}^n\right], \end{aligned} \tag{13}$$

where $M^n_{ij} = u^n_i - u^n_j$, if $i, j \geq n+1$ and $M^n_{ij} = 0$ otherwise.

It is possible to recover all the incompressible functions from Proposition 2 by adding the blocks $\mathbf{v}^0 + \mathbf{v}^1 + ... + \mathbf{v}^{d-2}$.

Proposition 3. *Let $\mathbf{v}$ be a divergence free function in $\mathbb{R}^d$. Then there exists $d-1$ functions $\mathbf{v}^0, ..., \mathbf{v}^{d-2}$ constructed as in Lemma 2 such that*

$$\mathbf{v}(\mathbf{x}) = \sum_{n=0}^{d-2} \mathbf{v}^n(\mathbf{x}).$$

The attentive reader would have noticed that with the vector functions $\mathbf{u}^n$, $n = 0, ..., d-2$ we have a total of $(d+2)(d-1)/2$ independent scalar functions while Proposition 2 only requires the construction of $d(d-1)/2$ scalar functions leaving $d-1$ additional functions which are not strictly needed to obtain the divergence free vectors. This is due to the vectorized constructions (12)-(13) which allow a fast evaluation of the divergence free functions on GPU. Having $d-1$ additional scalar functions in return is not a big issue since the general order remains $O(d^2)$.

Also note that the practical implementation of the equations (10)-(11) requires to find a pythonic way to efficiently pad a group of matrices with different dimensions. We have not yet find such way and therefore have simply chosen in our applications to construct $d-1$ vector valued functions $\mathbf{u}^n \in \mathbb{R}^d$, $n = 0, ..., d-2$ and fill the appropriate dimensions in (10)-(11) with 0. Again even if not optimal this is not a big issue as it multiplies the number of independent scalar functions by a factor 2, but the general order remains $O(d^2)$.

In practice, we have written the functions $\mathbf{u}^n$ as the output of a big function $\mathbf{u} : \mathbb{R}^d \to \mathbb{R}^{(d-1)\times d}$ allowing to evaluate all the functions $\mathbf{u}^n$ in a single pass. The vector $\mathbf{u}$ is written as the composition of linear functions with some simple non linearity

$$\begin{aligned} &\mathbf{u}(\mathbf{x}) = M_n\mathbf{x}_n + \mathbf{b}_n, \\ &\mathbf{x}_i = \sigma(M_{i-1}\mathbf{x}_{i-1} + \mathbf{b}_{i-1}), \quad i = 1, ..., n-1, \\ &\mathbf{x}_0 = \mathbf{x}, \end{aligned} \tag{14}$$

where M_i are rectangular matrices, $\mathbf{b}_i$ a vector field and typically we have taken $\sigma = \tanh$. One big advantage of the formulation (14) is that the Jacobian of $\mathbf{u}$ (and therefore of all the functions $\mathbf{u}^n$) can be computed analytically

$$\begin{aligned} &\nabla\mathbf{u}(\mathbf{x}) = M_n\nabla\mathbf{x}_n, \\ &\nabla\mathbf{x}_i = \operatorname{diag}(\sigma'(M_{i-1}\nabla\mathbf{x}_{i-1} + \mathbf{b}_{i-1}))M_{i-1}, \end{aligned} \tag{15}$$

for $i = 1, ..., n-1$. The formulation (15) therefore allows a fast evaluation of the term (12) in particular when summing the diagonal elements of the Jacobian. In our experiments we have noticed that the analytical formulation of the Jacobian (15) was faster than using *torch.autograd*.

C Experiment details

dSprites datasets. The VAE architecture used in the experiments is taken from the github repository of Yann Dubois[2]. The parameters used for the GP flows are given in Table 3. For all test cases we consider a GP flow with 15 time steps and three intermediate layers of 50 parameters with a Runge-Kutta 4 discretization at each layer. For the Euler penalization we take an initial parameter $\lambda = 5 \times 10^{-5}$ which is then divided 10 times periodically by a factor 2 during the training.

D Interpolation examples

[2] https://github.com/YannDubs/disentangling-vae

Model	# params (pre-trained model) + GP	epochs	batch size	lr
FFJORD+GP	(17.8K) + 10.3K	≈ 200*	1024	5×10^{-4}
FFJORD+GP+EULER	(17.8K) + 10.3K	3000	1024	5×10^{-4}

Table 3: Parameters used for the training of GP flows on the dSprites data set.
* The training is stopped early due to out-of-domain particles.

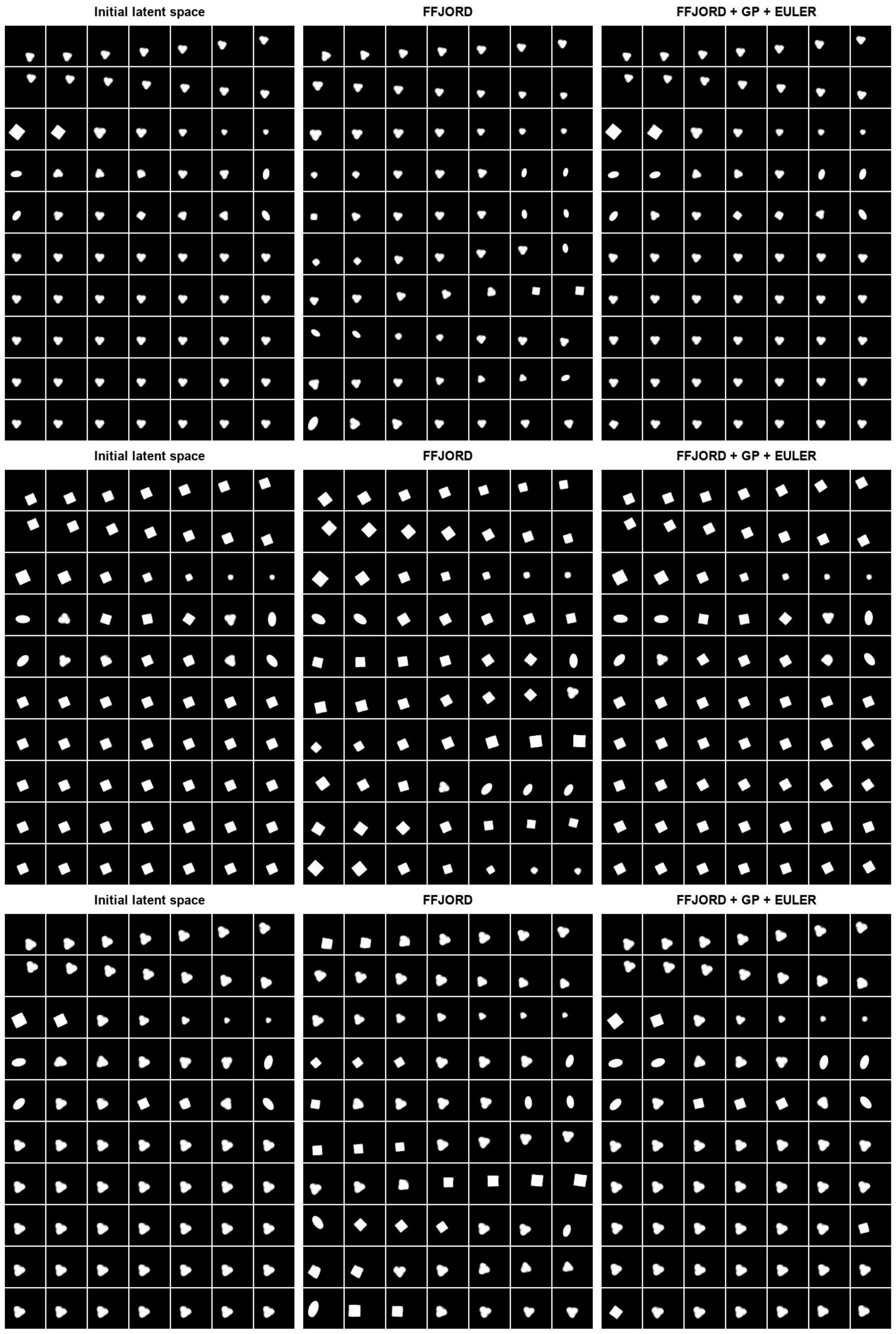

Figure 2: Examples of interpolation for the dSprites dataset where each block correspond to the interpolation of a different data point along the 10 dimensions axis represented by the rows. The dimensions are sorted with respect to their KL divergence in the VAE latent space, so the higher rows carry more information while the last rows should leave the image unchanged.

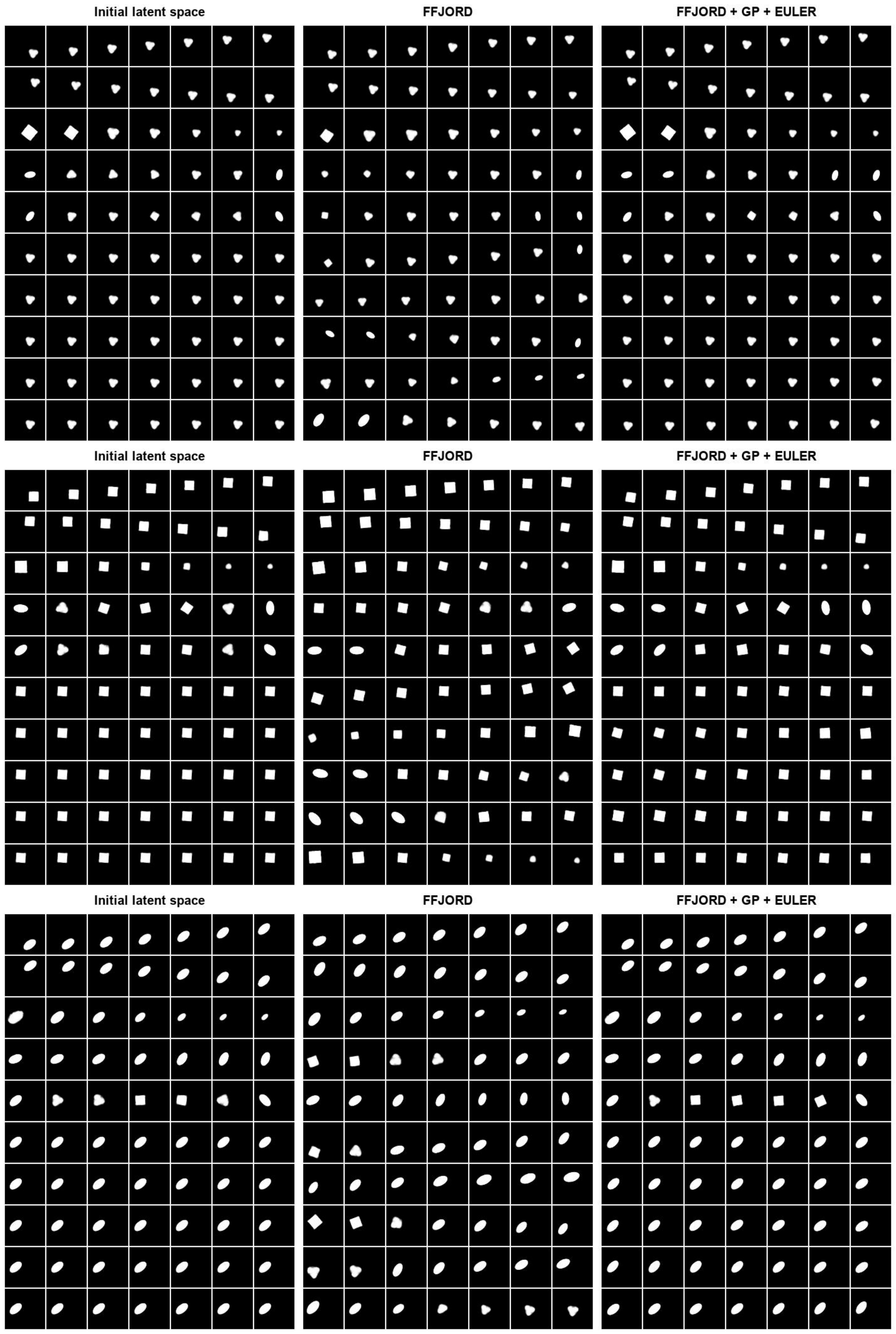

Figure 3: Examples of interpolation for the dSprites dataset where each block correspond to the interpolation of a different data point along the 10 dimensions axis represented by the rows. The dimensions are sorted with respect to their KL divergence in the VAE latent space, so the higher rows carry more information while the last rows should leave the image unchanged.