# Turning Normalizing Flows into Monge Maps with Geodesic Gaussian Preserving Flows

Guillaume Morel, Lucas Drumetz, Nicolas Courty, François Rousseau

# Turning Normalizing Flows into Monge Maps with Geodesic Gaussian Preserving Flows

**Guillaume Morel**[*] **Lucas Drumetz**[*] **Nicolas Courty**[†] **François Rousseau**[*]

## 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

Modeling high dimensional data is a central question in data science as they are ubiquitous in applications. Various tasks such as probabilistic inference, density estimation or sampling of new data require accurate probabilistic models that need to be defined efficiently. There exists a large variety of generative models in the literature. Among other approaches, variational autoencoders (VAES) (1; 2) and generative adversarial networks (GAN) (3) are frequent choices, each with their strengths and weaknesses.

**Normalizing flows.** A third 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 $\mu$ to the known source distribution $\nu = \mathbf{f}_{\#}\mu$ (4; 5). In practice the flow must satisfy the change of variables formula:

$$\log p_\mu(\mathbf{x}) = \log p_\nu(\mathbf{f}(\mathbf{x})) + \log |\det \nabla \mathbf{f}(\mathbf{x})|. \tag{1}$$

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. The data are generally high dimensional and accessing $p_\mu(\mathbf{x})$ for a given $\mathbf{x}$ requires computing determinant of the Jacobian matrix of $\mathbf{f}$. This operation has a complexity of $O(d^3)$ in general and thus a requirement for the flows architecture is to have a tractable determinant of the Jacobian while remaining expressive enough (4; 6; 5; 7).

**Optimal transport.** A diffeomorphism transforming any well-behaved distribution into another always exists in theory (7). However, there can be many ways to transform one probability measure $\mu$ into another probability measure $\nu$, and therefore the function $\mathbf{f}$ is generally not unique. This has led to many proposed architectures in the literature (6; 8; 9; 10; 11). The question of choosing the "best" transformation among all existing ones is therefore crucial, independently from how accurately $\mu$

[1] IMT Atlantique: {guillaume.morel, lucas.drumetz, francois.rousseau}@imt-atlanique.fr [2] IRISA: nicolas.courty@irisa.fr

Preprint.

models the target distribution. One way to make the architecture unique (under appropriate conditions on the two distributions) is to use optimal transport (12; 13; 14; 15), that is to choose the one giving the Wasserstein distance between $\mu$ and $\nu$, with a squared $\mathcal{L}_2$ ground cost:

$$W_2^2(\mu,\nu) = \min_{\mathbf{f}} \int_{\mathbb{R}^d} |\mathbf{f}(\mathbf{x}) - \mathbf{x}|^2 d\mu(\mathbf{x}), \quad \nu = \mathbf{f}_{\#}\mu. \tag{2}$$

An optimal model in the sense of (2) minimizes the total mass displacement which can be a desirable property even if it is often a difficult task. In particular Brenier's theorem (16) states that the optimal function $\mathbf{f}$ is the gradient of a scalar convex function, which is widely used when solving (2).

One key property of OT mappings is that they should better preserve the structure of the distribution compared to non OT transformations. This makes them particularly appealing for machine learning applications, and may also help with generalization performance (17).

**Optimal transport and NF models.** Including OT in NF models has recently received much attention with various approaches to obtain a map $\mathbf{g}$ which satisfies the property (2). Among all these methods, many use either directly Brenier's theorem (16) or the dynamic OT formulation with the Benamou-Brenier approach (18). One important remark is that most of the approaches considered need dedicated architectures in order to satisfy the OT property. For example the transformation $\mathbf{f}$ is often written as neural network modeling the gradient of a (possibly convex) scalar function (19; 20; 21; 22). This sometimes requires some particular training process (19; 20; 21) and/or the addition of some penalization terms in the loss function (21; 23; 24). When considering the Benamou-Brenier formulation, the normalizing flow is interpreted as the discretization of a continuous ordinary differential equation (25) and the optimal transport problem is then solved dynamically (23; 21; 22).

### 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 $\nabla\psi$ solving (2) 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 $\mu$ to $\nu$ and $\mathbf{s}$ an associated measure preserving function of $\nu$ (16). 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 $\nu$ 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 $\nu$ is known and simple (here and in the following $\nu$ is a standard normal) which make it possible to construct architectures preserving $\nu$.

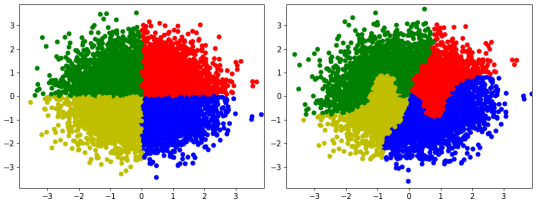

**Figure 1:** A GP transformation applied on particles sampled from a two dimensional normal distribution. The mean and standard deviation stay the same; only the positions of the particles change.

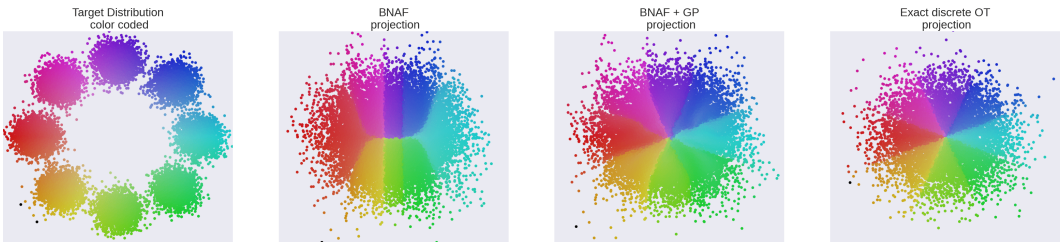


**Figure 2:** Eight gaussians test case with colored distributions. A GP flow is trained on a pre-trained BNAF model (10) to reduce the OT cost.

Our work differs from the state of the art as we do not propose a new normalizing flow model. Instead we propose to use Brenier's theorem to compute the Monge map for any pre-existing architecture.

Indeed there exists a wide variety of architecture available in the literature (6; 8; 9; 10; 11) each with their pros and cons which sometimes depend specifically on the test case considered. Our idea is to use Brenier's polar factorization theorem to rearrange the points in the known distribution to obtain the optimal map associated with a given flow. We consider the most common case where the known distribution is a standard normal and call such rearranging maps Gaussian Preserving (GP) flows, see Figure 1. 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. This 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 2.

**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. This is somehow similar to the approaches from (23; 21) where the trajectories of a continuous normalizing flow are penalized to be straight lines. This is not strictly needed to find the Monge map but can be interpreted as some geodesic over all the flows which solve the associated OT problem. Such an approach has already shown some promising results. 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 (26). The penalization of Euler's equations in high dimensions and its practical implementation is therefore also considered, which is to the best of our knowledge an original contribution.

**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 (27) 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 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 $\nu$. To preserve the conventions from Brenier's paper (16), we study the OT problem defined from the known probability distribution $\nu$ to $\mu$ and therefore consider the function $\mathbf{g} := \mathbf{f}^{-1}$.

**Theorem 1** (Brenier's polar factorization (16))**.** *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 $\nu$ 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.

**Remark 1.** *Since in practice we consider $\nu$ to be a standard normal, the domain $\mathcal{X}$ is not bounded and therefore does not strictly satisfy the hypothesis of Theorem 1. We do not investigate this point further and simply quote a remark from Brenier's work (16): "we believe that the result is still true when $\mathcal{X}$ is unbounded, provided that $p > 1$ and $\int_{\mathcal{X}} \|\mathbf{x}\|^q \beta(\mathbf{x}) d\mathbf{x} < +\infty$, where $1/q + 1/p = 1$". The function $\beta(\mathbf{x}) = e^{\|\mathbf{x}\|^2/2}$ is the probability density of $\nu$, and the inequality is therefore satisfied.*

## 3 Gaussian preserving flows

In order to apply Brenier's polar factorization theorem, it is therefore needed to construct a class of measure preserving maps. Since we consider the case where $\nu$ is a standard normal, we call such maps Gaussian preserving (GP). All proofs of the propositions and lemmas are given in Appendix B.

Consider two probability measures $\mu$ and $\nu$ with density $h_\mu$ and $h_\nu$ respectively. A map $\mathbf{s}$ is measure preserving between $\mu$ and $\nu$ if it satisfies the change of variable equality (same as (1) without the log) $h_\nu(\mathbf{x}) = h_\mu(\mathbf{s}(\mathbf{x}))|\det(\nabla\mathbf{s}(\mathbf{x}))|$. In our case, we want $\mathbf{s}$ to be Gaussian preserving therefore

$h_\mu = h_\nu = e^{-\|\mathbf{x}\|^2/2}$ and one gets

$$|\det \nabla \mathbf{s}(\mathbf{x})| = e^{(\|\mathbf{s}(\mathbf{x})\|^2 - \|\mathbf{x}\|^2)/2}. \tag{3}$$

It turns out that Lebesgue preserving functions (i.e. satisfying $|\det \nabla \boldsymbol{\phi}| = 1$) can be used to construct maps satisfying (3). 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 Gaussian preserving function (i.e. satisfying* (3)*). 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. In Appendix B.2 we also 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.

### 3.1 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 $\Omega$ then

$$\mathrm{SDiff}(\Omega) := \big\{ \boldsymbol{\psi} \in \mathrm{Diff}(\Omega),\ \det(\nabla \boldsymbol{\psi})(\mathbf{x}) = 1,\ \forall \mathbf{x} \in \Omega \big\},$$

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:

$$\begin{cases} \frac{d}{dt}\mathbf{X}(t,\mathbf{x}) = \mathbf{v}(t, \mathbf{X}(t,\mathbf{x})), & \mathbf{x} \in \Omega, \quad 0 \le t \le T, \\ \mathbf{X}(0,\mathbf{x}) = \mathbf{x}. \end{cases} \tag{4}$$

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

$$\nabla \cdot \mathbf{v} = 0, \quad \text{in } \Omega, \tag{5}$$

$$\mathbf{v} \cdot \mathbf{n} = 0, \quad \text{on } \partial\Omega, \tag{6}$$

where $\mathbf{n}$ is the outward normal at the boundary of $\Omega$. We define $\boldsymbol{\phi}$ to be the solution at the final time $\boldsymbol{\phi}(\mathbf{x}) := \mathbf{X}(T, \mathbf{x})$. Property (5) implies that $\det \nabla \boldsymbol{\phi} = 1$, and property (6) ensures that $\boldsymbol{\phi}$ does not escape $\Omega$. Any function in $\mathrm{SDiff}(\Omega)$ can be written as a solution to (4) for $d \ge 3$ (28), for $d = 2$ some pathological cases can be constructed (29).

**Divergence free vector fields.** First we focus on the vector fields satisfying (5) for arbitrary large dimensions. Property (6) can then be incorporated with very little additional work.

**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*

$$v_i(\mathbf{x}) = \sum_{j=1}^{d} \partial_{x_j} \psi^i_j(\mathbf{x}), \quad i = 1, ..., d, \tag{7}$$

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

To impose the boundary conditions (6) one can simply multiply each $\psi^i_j$ 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 incompressible property (5) and the boundary conditions (6) can be exactly implemented in the network in any dimensions. Note however that in order to get all the incompressible vector fields (7), we need to construct $d(d-1)/2$ arbitrary scalar functions. See Appendix A for the practical contruction of these divergence free functions.

# 4 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.

## 4.1 Arnold's theorem

In 1966, Arnold (26) 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:

$$\begin{cases} \partial_t \mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\nabla p, & t \in [0,T],\ \mathbf{x} \in \Omega, \\ \nabla \cdot \mathbf{v} = 0, & t \in [0,T],\ \mathbf{x} \in \Omega, \\ \mathbf{v} \cdot \mathbf{n} = 0, & t \in [0,T],\ \mathbf{x} \in \partial\Omega, \\ \mathbf{v}(0,\cdot) = \mathbf{v}_0, \end{cases} \tag{8}$$

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 $\Omega$. We introduce $\mathcal{E}$ the energy of a smooth function $\mathbf{X}(t,\cdot)$:

$$\mathcal{E}(\mathbf{X}) = \int_0^T \int_\Omega \frac{1}{2} |\partial_t \mathbf{X}(t,\mathbf{x})|^2 d\mathbf{x} dt, \tag{9}$$

Now assume $\boldsymbol{\phi} \in \mathrm{SDiff}(\Omega)$. Arnold's problem's consists in finding the path $\mathbf{X}(t,\cdot)_{t\in[0,T]}$ in $\mathrm{SDiff}(\Omega)$ joining the identity to $\boldsymbol{\phi}$ which minimizes $\mathcal{E}$:

$$\min_{\mathbf{X}(t,\cdot)\in \mathrm{SDiff}(\Omega)} \mathcal{E}(\mathbf{X}), \quad \mathbf{X}(0,\cdot) = \mathrm{Id}, \quad \mathbf{X}(T,\cdot) = \boldsymbol{\phi}(\cdot). \tag{10}$$

In other words (10) is the geodesic in $\mathrm{SDiff}(\Omega)$ between Id and $\boldsymbol{\phi}$.

**Theorem 2** (Arnold (26))**.** *Assuming the existence of a solution to Arnold's problem,* $\mathbf{X}$ *is solution to* (10) *if and only if* $\mathbf{v}(t,\mathbf{x}) := \partial_t \mathbf{X}(t,\mathbf{x})$ *satisfies Euler's equations* (8)*.*

## 4.2 Penalization of Euler's equations in high dimensions

Numerical schemes developed to efficiently solve the Euler equations (30; 31) (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 (8) through a penalization procedure which can be carried out in any dimensions. As explained in the previous section we notice that the second and third equations in (8) 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 (8), 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:

$$R(\mathbf{x}) := \mathbb{E}_{\mathbf{y},\mathbf{z}} \left[ \int_0^T \left( \mathbf{y}^T (\nabla \mathbf{w}_{t,\mathbf{x}}) \mathbf{z} - \mathbf{z}^T (\nabla \mathbf{w}_{t,\mathbf{x}}) \mathbf{y} \right)^2 dt \right], \quad \mathbf{y}, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathrm{Id}), \tag{11}$$

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 (11) 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 (4) that is the position of a particle at time $t$ with initial position $\mathbf{x}$. We recall the equality (see Appendix B.5) $\frac{D}{Dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) = \partial_t \mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) + (\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) \cdot \nabla)\mathbf{v}(t, \mathbf{X}(t, \mathbf{x}))$ and therefore choose to approximate the right hand side by using a first order Taylor expansion of $D\mathbf{v}/Dt$:

$$\frac{D}{Dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) \approx \frac{\mathbf{v}(t^{n+1}, \mathbf{X}(t^{n+1}, \mathbf{x})) - \mathbf{v}(t^n, \mathbf{X}(t^n, \mathbf{x}))}{\Delta t}, \quad \Delta t := t^{n+1} - t^n. \tag{12}$$

In practice, $\Delta t$ is set to $2\sqrt{\varepsilon}$ where $\varepsilon$ 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.

## 5 Procedure

In order to solve the optimal transport problem, we consider the inverse of the transformation given in Theorem 1 (that is $\mathbf{f} := \mathbf{g}^{-1}$) because the NF architectures are defined in practice from the unknown probability distribution $\mu$ to a Gaussian distribution $\nu$ and are not always easily invertible.

The GP flow is parametrized as a standard residual network (ResNet) with a Runge-Kutta 4 time discretization (32) (other discretizations are possible) and is estimated by minimizing $\|\mathbf{x} - \mathbf{s} \circ \mathbf{f}(\mathbf{x})\|^2$ over the parameters of the velocity field. In practice when regularizing with Euler's equations, we replace the term $\nabla \mathbf{w}$ in (11) by (12) and calculate the Jacobian-vector product with the function *torch.autograd* from pytorch. A parameter $\lambda > 0$ is also added in front of the penalization term:

$$\min_{\boldsymbol{\theta}} E_{\mu(\mathbf{x})} \left[ \|\mathbf{x} - \mathbf{s}_{\boldsymbol{\theta}} \circ \mathbf{f}(\mathbf{x})\|^2 + \lambda R_{\boldsymbol{\theta}} \circ \mathbf{erf} \circ \frac{\mathbf{f}(\mathbf{x})}{\sqrt{2}} \right], \tag{13}$$

where the vector $\boldsymbol{\theta}$ denotes the parameters of the velocity field $\mathbf{v}$, $\mathbf{f}$ is the NF architecture, $\mathbf{s}$ the GP flow and $R$ corresponds to the term penalized with Euler's equations. The subscript $\boldsymbol{\theta}$ has been added to emphasize the dependence of $\mathbf{s}$ and $R$ to the parameters. To minimize (13), a sampling strategy from the probability distribution $\mu$ is required. Two possibilities can be considered:

- If we do not want to invert the NF model (for example if it is computationally expensive) we can simply minimize the first term in (13) over the training data. The probability distribution $\mu$ in (13) is then the unknown distribution from which the data are taken. This assumes however that the training data are correctly mapped to the standard distribution $\nu$ with $\mathbf{f}$.
- If the NF model is cheap to invert, we consider the points $\mathbf{f}^{-1}(\mathbf{x}_G)$ where $\mathbf{x}_G$ are samples from the standard normal distribution $\nu$. In this case, the probability distribution $\mu$ in (13) is the transformation of $\nu$ by the inverse of the NF model $\mu = \mathbf{f}^{-1}_{\#}\nu$.

If the NF model transforms perfectly the training data over the standard normal $\nu$, these two approaches are equivalent. If this is not the case, we notice that the second approach requires a cheap inverse of the NF model, but has the advantage of not using any training data to train the GP flow and may therefore better generalize especially when the training data does not match the Gaussian well.

## 6 Results

We perform density estimation and evaluate the OT cost on two dimensional toy problems and real high dimensional tabular data sets. We apply GP flows on two popular NF models: FFJORD, a continuous NF (8) and BNAF, a discrete NF (10). Both of these models are solid references among NF and do not incorporate any OT knowledge in their architecture or training procedure. To compare our results with an OT-efficient approach, we also consider the CP-Flow architecture (20). The CP-flow network is constrained to be the gradient of a scalar convex function and converges by construction towards the optimal map (provided the optimization problem reaches a global minimum), making it a good candidate for comparison. All the codes are taken from the official repositories[1]. The FFJORD model has an inverse function directly available in the code, which is not the case for the BNAF model. Therefore, as explained in the previous section, the GP flow is trained using $\mathbf{f}^{-1}$ applied on a standard multi-dimensional normal distribution for FFJORD, whereas only the training data are used for BNAF. We will consider the BNAF model in 2D and the FFJORD model on the dSprites dataset ($d = 10$) because interpolations in the dSprites latent space require the NF architecture to have an inverse function available.

[1] github.com/rtqichen/ffjord, github.com/nicola-decao/BNAF, github.com/CW-Huang/CP-Flow

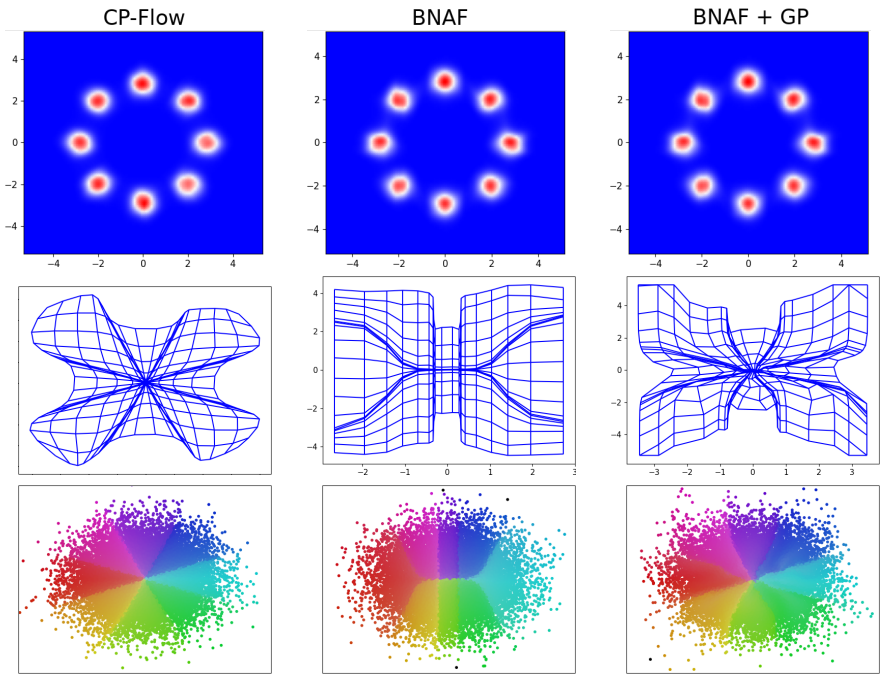


**Figure 3:** Eight gaussians test case. Top: density estimation. Middle: deformation of a uniform mesh by the NF model. Bottom: Colored source distribution. From left to right: CP-Flow (OT=2.62, loss=2.86), BNAF (OT=2.88, loss=2.85), BNAF+GP (OT=2.60, loss=2.85).

### 6.1 Density estimation on toy 2D data

We perform density estimation on several 2d standard toy distributions (8; 33). First we consider the eight gaussians test case and study the transformation of a uniform mesh by the NF model. We run the CP-Flow and the BNAF model. The GP transformation is then computed on the pre-trained BNAF models. As shown in Figure 3, the mesh transformation when adding the GP flow is getting closer to the CP-Flow one, and the OT cost is roughly the same. Another illustration of the transformation of the source distribution is given at the bottom of Figure 3. We clearly see the added value of GP flow, as the points' configuration gets much closer to an isotropic distribution of the colors on the Gaussian, that indicates that we get much closer to the Monge map. Note that adding the GP flow does not affect the estimated density nor the test loss.

**Euler's penalization.** To highlight the value of using the Euler's penalization we turn our attention to the two moons dataset. First, to evaluate the quality of the solution of the Euler equations with the proposed approach, we compare our solution with a more standard approach where the initial condition is optimized through a numerical scheme (implemented in Pytorch, so as to be differentiable) which directly solves the Euler equations, in the line of "differentiable physics" approaches (34). The baseline numerical scheme is a spectral method, a very efficient and popular numerical method (30). We do not go into details on how to implement such methods since they are restricted in practice to low dimensions and we refer to (30; 31) for a complete presentation. On Figure 4, we compare three different cases: GP alone, GP with Euler penalization and GP with Euler constrain through a spectral method. As indicated by the similar OT costs the final positions of particles are very similar for each model. Turning our attention to the trajectories, the initial incompressible velocities and the energy $\bar{\mathcal{E}}$ are very close both for the penalization-based procedure and the spectral method which shows that we can correctly solve the Euler equations in 2D with our penalization approach (the additional benefit being the generalization to high dimensions). Without the addition of the Euler constraint however, the initial velocity field looks very different resulting in a higher energy, and thus a transformation that does not correspond to a geodesic in $\mathrm{SDiff}(\Omega)$.

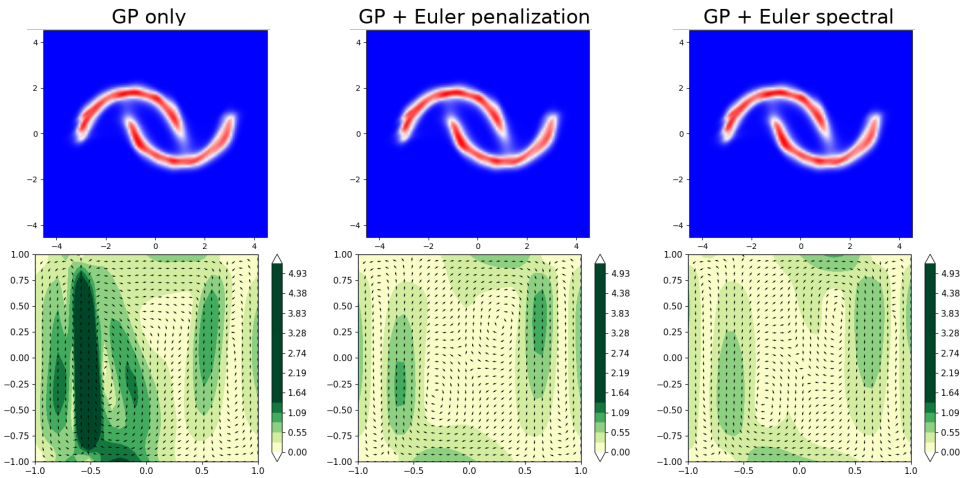


**Figure 4:** Two moons test case with the BNAF model (initial OT=1.35). Top: target density after the application of the GP flow. Bottom: representation of the initial incompressible velocity field $\mathbf{v}_0$ of the GP flow. From left to right: GP only ($\bar{\mathcal{E}} = 0.2$, OT=0.99), GP with Euler through the penalization procedure (11) ($\bar{\mathcal{E}} = 0.08$, OT=1.0), GP with a spectral method solving directly the Euler equations ($\bar{\mathcal{E}} = 0.07$, OT=1.0). For the last two mentioned the initial velocity fields are the same and the energy $\bar{\mathcal{E}}$ is lowered showing that the penalization procedure efficiently solve Euler's equations.

### 6.2 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) (1) with disentangle latent space has received much attention (35; 36; 37; 38). 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 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 $\beta$-TCVAE architecture (37) and the latent space dimension is 10.

**dSprites test case.** The dSprites dataset (27) is made of $64 \times 64$ 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 (39) 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 5 and 6 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

| 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) on the dSprites dataset. 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 and mean OT costs for the dSprites dataset. GP flows reduce the OT cost without changing the loss. Adding Euler regularization allows to further reduced the OT cost.

each of the first 5 lines correspond to a generative factor. 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.

## References


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

[2] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

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

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

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

[7] 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.

[8] 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.

[9] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR, 2018.

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

[11] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

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

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

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

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

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

[17] Skander Karkar, Ibrahim Ayed, Emmanuel de Bézenac, and Patrick Gallinari. A principle of least action for the training of neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 101–117. Springer, 2020.

[18] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84:375–393, 2000.

[19] Chris Finlay, Augusto Gerolin, Adam M Oberman, and Aram-Alexandre Pooladian. Learning normalizing flows from entropy-kantorovich potentials. *arXiv preprint arXiv:2006.06033*, 2020.

[20] Chin-Wei Huang, Ricky TQ Chen, Christos Tsirigotis, and Aaron Courville. Convex potential flows: Universal probability distributions with optimal transport and convex optimization. In *International Conference on Learning Representations*, 2020.

[21] Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. 2020.

[22] Linfeng Zhang, Lei Wang, et al. Monge-ampere flow for generative modeling. *arXiv preprint arXiv:1809.10188*, 2018.

[23] Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *International conference on machine learning*, pages 3154–3164. PMLR, 2020.

[24] Liu Yang and George Em Karniadakis. Potential flow generator with l2 optimal transport regularity for generative models. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[25] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[26] 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.

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

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

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

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

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

[32] Kendall E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, New York, second edition, 1989.

[33] Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in neural information processing systems*, 32, 2019.

[34] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.

[35] 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.

[36] 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.

[37] 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.

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

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

# A Practical construction of incompressible vector fields in high dimensions

The goal here is to construct incompressible vector fields as in Proposition 2.

## A.1 A first approach

Let $\mathbf{u} : \mathbb{R}^d \to \mathbb{R}^d$ be a vector valued function. This time we consider the case

$$\psi_j^i = u_i - u_j.$$

Note that even if this is more general than the first case the functions $\psi_j^i$ satisfy the relation

$$\psi_j^i = \psi_k^i + \psi_j^k \tag{14}$$

and therefore do not describe completely the divergence-free functions from Proposition 2.

**Lemma 2.** *Let* $\mathbf{u} : \mathbb{R}^d \to \mathbb{R}^d$ *then the vector field*

$$\mathbf{v}(\mathbf{x}) = \nabla \mathbf{u}\, \mathbf{1} - \mathrm{diag}(\nabla \mathbf{u})\, \mathbf{1}, \tag{15}$$

*is divergence free. Here* $\mathbf{1}$ *is the unit vector* $\mathbf{1} = (1, ..., 1)$.

To construct these functions 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. To satisfy the boundary conditions one can modify this equation as in Lemma 1 to obtain

$$\begin{gathered} \mathbf{v}(\mathbf{x}) = (\mathbf{x}^2 - 1) \odot \\ \left[2M\mathbf{x} + \nabla \mathbf{u}(\mathbf{x})(\mathbf{x}^2 - 1) - ((\mathbf{x}^2 - 1) \cdot \mathrm{diag}\, \nabla \mathbf{u}(\mathbf{x}))\mathbf{1}\right], \end{gathered} \tag{16}$$

where $M \in \mathbb{R}^{d \times d}$ is the matrix with the components $M_{ij} = u_i - u_j$.

In practice we have written the function $\mathbf{u}$ as the composition of linear functions with some simple non linearity

$$\begin{aligned}&\mathbf{u}(\mathbf{x}) = M_n\mathbf{x}_n + \mathbf{b}_n,\\ &\mathbf{x}_i = \sigma(M_{i-1}\mathbf{x}_{i-1} + \mathbf{b}_{i-1}), \quad i = 1, ..., n-1,\\ &\mathbf{x}_0 = \mathbf{x},\end{aligned} \tag{17}$$

where $M_i$ are rectangular matrices and $\mathbf{b}_i$ vector field and typically we have taken $\sigma = \tanh$. The advantage with the formulation (17) is that the Jacobian can be computed analytically

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

for $i = 1, ..., n-1$. This allows a fast evaluation of the term (16) in particular when summing the diagonal elements of the Jacobian.

### A.2 Generalization to all incompressible vector fields

As stated above the functions (16) do not describe all the incompressible vector fields from Proposition 2 due to the equality (14). In practice it should be possible to enriched this space by considering functions $\mathbf{u}^n : \mathbb{R}^d \to \mathbb{R}^{d-n}$ and then padded the missing dimensions with zeros. This is equivalent to consider

$$\psi_j^i = \begin{cases} u_i^n - u_j^n, & \text{if } i, j \geq n+1,\\ 0, & \text{otherwise.}\end{cases}$$

To construct our solutions based on $\mathbf{u}^n$ 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

$$(\nabla\mathbf{u}(\mathbf{x}))_{ij}^n = \begin{cases}\partial_i u_j^n, & \text{if } i, j \geq n+1\\ 0, & \text{otherwise.}\end{cases}, \qquad \mathbf{1}_i^n = \begin{cases}1, & \text{if } i \geq n+1\\ 0, & \text{otherwise.}\end{cases}.$$

**Lemma 3.** *Let $n \in \mathbb{N}^*$, $n \leq d-1$ 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*

$$\mathbf{v}^n(\mathbf{x}) = (\nabla\mathbf{u})^n\ \mathbf{1}^n - \operatorname{diag}(\nabla\mathbf{u})^n\ \mathbf{1}^n, \tag{18}$$

*is divergence free.*

To satisfy the boundary condition one can modify the equation (18) as in Lemma 1 to obtain

$$\begin{aligned}&\mathbf{v}^n(\mathbf{x}) = (\mathbf{x}^2 - 1)\odot\\ &\left[2M^n\mathbf{x} + (\nabla\mathbf{u})^n(\mathbf{x})(\mathbf{x}^2-1) - ((\mathbf{x}^2-1)\cdot\operatorname{diag}(\nabla\mathbf{u}(\mathbf{x}))^n)\mathbf{1}^n\right],\end{aligned} \tag{19}$$

where $M_{ij}^n = u_i^n - u_j^n$, if $i, j \geq n+1$ and $M_{ij}^n = 0$ otherwise.

One could then add some functions $\mathbf{v}^0 + \mathbf{v}^1 + \mathbf{v}^2$... to get a more general incompressible function (the approach given in (16) is equivalent to consider only $\mathbf{v}^0$). It is possible to recover all the incompressible functions from Proposition 2 by adding the blocks $\mathbf{v}^0 + \mathbf{v}^1 + ... + \mathbf{v}^{d-2}$ because the problematic relation (14) is broken thanks to the zeros padding of the new blocks. Note that in practice this requires to construct the $d-1$ vector valued functions $\mathbf{u}^n \in \mathbb{R}^{d-n}$, $n = 0, ..., d-2$, and therefore to find a pythonic way to efficiently pad matrices with different dimensions. For simplicity, we have chosen to only construct $d-1$ vector valued functions $\mathbf{v}^n \in \mathbb{R}^d$ in our applications.

## B Technical material

### B.1 Proof of Proposition 1

We recall that here $\mathbf{erf} : \mathbb{R}^d \to \mathbb{R}^d$ is the distribution function of a one dimensional Gaussian applied component wise.

**Proposition.** *Let $\mathbf{s}$ a smooth Gaussian preserving function satisfying* (3)*. Then there exists $\boldsymbol{\phi} : (-1,1)^d \to (-1,1)^d$ such that $|\det\nabla\boldsymbol{\phi}| = 1$ and*

$$\mathbf{s}(\mathbf{x}) = \sqrt{2}\,\mathbf{erf}^{-1}\circ\boldsymbol{\phi}\circ\mathbf{erf}(\frac{\mathbf{x}}{\sqrt{2}}), \quad \mathbf{x}\in\mathbb{R}^d. \tag{20}$$

*Proof.* We recall some basic properties about the distribution function of a one dimensional Gaussian and its inverse. One has for $x \in \mathbb{R}$

$$\begin{aligned} \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{t^2} dt, \quad \frac{d}{dx} \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} e^{-x^2}, \\ \frac{d}{dx} \operatorname{erf}^{-1}(x) = \frac{\sqrt{\pi}}{2} e^{(\operatorname{erf}^{-1}(x))^2}. \end{aligned} \tag{21}$$

Consider the function $\boldsymbol{\phi} : (-1,1)^d \to (-1,1)^d$ defined as

$$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{erf} \circ \frac{\mathbf{s}}{\sqrt{2}} \circ \sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x}). \tag{22}$$

The goal here is to show that $|\det \nabla \boldsymbol{\phi}| = 1$ then equation (20) will follows from (22). By definition

$$|\det \nabla \boldsymbol{\phi}(\mathbf{x})| := |\det \nabla \left( \mathbf{erf} \circ \frac{\mathbf{s}}{\sqrt{2}} \circ \sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x}) \right)|.$$

Applying the equalities (21) component wise and denoting $\mathbf{x} = (x_1, ..., x_d)$ one gets

$$\begin{aligned} |\det \nabla \boldsymbol{\phi}(\mathbf{x})| = |\prod_i \frac{2}{\sqrt{\Pi}} e^{-(\frac{\mathbf{s}}{\sqrt{2}} \circ \sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x}))_i^2} \times \frac{\det \nabla \mathbf{s}(\sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x}))}{\sqrt{2}} \\ \times \prod_i \sqrt{2} \frac{\sqrt{\Pi}}{2} e^{(\mathbf{erf}^{-1}(\mathbf{x}))_i^2}|, \end{aligned}$$

since the determinant of the composition is the product of the determinants. That is

$$\begin{aligned} |\det \nabla \boldsymbol{\phi}(\mathbf{x})| = |\prod_i e^{-(\frac{\mathbf{s}}{\sqrt{2}} \circ \sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x}))_i^2} \times \det \nabla \mathbf{s}(\sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x})) \\ \times \prod_i e^{(\mathbf{erf}^{-1}(\mathbf{x}))_i^2}|, \end{aligned}$$

which can be written

$$|\det \nabla \boldsymbol{\phi}(\mathbf{x})| = e^{(-\|\mathbf{s}(\sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x}))\|^2 + \|\sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x})\|^2)/2} \times |\det \nabla \mathbf{s}(\sqrt{2}\, \mathbf{erf}^{-1}(\mathbf{x}))|.$$

Finally using $|\det \nabla \mathbf{s}(\mathbf{x})| = e^{(\|\mathbf{s}(\mathbf{x})\|^2 - \|\mathbf{x}\|^2)/2}$ one obtains $|\det \nabla \boldsymbol{\phi}(\mathbf{x})| = 1$. □

### B.2 Orientation reversing function

In the following Lemma we prove that very orientation reversing function satisfying $\det \nabla \boldsymbol{\psi} = -1$ can be written as the composition of a volume and orientation preserving function and the function $\boldsymbol{h}(x_1, ..., x_d) = (-x_1, x_2, x_3, ..., x_d)$.

**Lemma.** *Assume $\boldsymbol{\psi} : \mathbb{R}^d \to \mathbb{R}^d$ is a function satisfying $\det \nabla \boldsymbol{\psi} = -1$ and let $\boldsymbol{h}(x_1, ..., x_d) = (-x_1, x_2, x_3, ..., x_d)$. Then there exists a volume and orientation preserving function $\boldsymbol{\phi}$ such that $\boldsymbol{\psi} = \boldsymbol{\phi} \circ \boldsymbol{h}$.*

*Proof.* We define $\boldsymbol{\phi}$ as $\boldsymbol{\phi} = \boldsymbol{\psi} \circ \boldsymbol{h}$. This function is indeed volume and orientation preserving since it satisfies $\det \nabla \boldsymbol{\phi} = \det \nabla \boldsymbol{\psi} \det \nabla \boldsymbol{h} = 1$. By noticing $\boldsymbol{h} \circ \boldsymbol{h} = \mathrm{Id}$ one gets $\boldsymbol{\psi} = \boldsymbol{\phi} \circ \boldsymbol{h}$. □

**Lemma.** *Assume the Monge map $\mathbf{m}$ and the NF architecture $\mathbf{g}$ are $C^1$ diffeomorphisms. Then the corresponding GP flow $\mathbf{s}$ is $C^1$, the associated function $\boldsymbol{\phi}$ is also $C^1$ and either satisfies $\det \nabla \boldsymbol{\phi}(\mathbf{x}) = 1 \; \forall \, \mathbf{x}$ or $\det \nabla \boldsymbol{\phi}(\mathbf{x}) = -1 \; \forall \, \mathbf{x}$.*

*Proof.* The definition of $\mathbf{s} := \mathbf{m} \circ \mathbf{g}^{-1}$ ensures that $\mathbf{s}$ is indeed $C^1$. Moreover since $\mathbf{g}$ is invertible either $\det \nabla \mathbf{g} > 0$ everywhere or $\det \nabla \mathbf{g} < 0$ everywhere (if $\det \nabla \mathbf{g}(x) = 0$ this would means that g is not invertible at this point) and the same argument applies to $\mathbf{g}^{-1}$ and to the Monge map $\mathbf{m}$. Therefore the equality $\mathbf{s} = \mathbf{m} \circ \mathbf{g}^{-1}$ implies that either $\det \nabla \mathbf{s} > 0$ everywhere or $\det \nabla \mathbf{s} < 0$ everywhere. The equality $\boldsymbol{\phi} = \mathbf{erf} \circ \frac{\mathbf{s}}{\sqrt{2}} \circ \sqrt{2}\, \mathbf{erf}^{-1}$ shows that $\boldsymbol{\phi}$ is also $C^1$ and that the sign of $\det \nabla \boldsymbol{\phi}$ does not change. □

### B.3 Proof of Proposition 2

**Proposition.** *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*

$$v_i(\mathbf{x}) = \sum_{j=1}^{d} \partial_{x_j} \psi^i_j(\mathbf{x}), \quad i = 1, ...d, \tag{23}$$

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

*Proof.* 1) First we prove that $\nabla \cdot \mathbf{v} = 0$. The divergence of $\mathbf{v}$ can be written

$$\nabla \cdot \mathbf{v} = \sum_i \partial_{x_i} \sum_j \partial_{x_j} \psi^i_j.$$

Using $\psi^i_j = -\psi^j_i$ one has

$$\nabla \cdot \mathbf{v} = \sum_i \left( \sum_{\substack{j \\ j<i}} \partial_{x_i} \partial_{x_j} \psi^i_j - \sum_{\substack{j \\ j>i}} \partial_{x_i} \partial_{x_j} \psi^j_i \right), \tag{24}$$

For the term $\sum_i \sum_{\substack{j \\ j<i}} \partial_{x_i} \partial_{x_j} \psi^i_j$ on the left hand side one can sum over the index $j$ first instead of the index $i$ that is

$$\sum_i \sum_{\substack{j \\ j<i}} \partial_{x_i} \partial_{x_j} \psi^i_j = \sum_j \sum_{\substack{i \\ i>j}} \partial_{x_i} \partial_{x_j} \psi^i_j.$$

Injecting this equality in (24) one gets

$$\nabla \cdot \mathbf{v} = \sum_j \sum_{\substack{i \\ i>j}} \partial_{x_i} \partial_{x_j} \psi^i_j - \sum_i \sum_{\substack{j \\ j>i}} \partial_{x_i} \partial_{x_j} \psi^j_i = 0.$$

2) Now we prove that every vector field satisfying $\nabla \cdot \mathbf{v} = 0$ can be written under the form (23). The proof from Stephen Montgomery-Smith is available online[2] for completeness we rewrite it here. The proof is made by induction with the following assumption.

**Assumption 1.** *Let* $k \in \mathbb{N}$*,* $k \leq d$*. Given a smooth vector field* $\mathbf{v}$ *such that* $\operatorname{div}_k \mathbf{v} := \sum_{i=1}^{k} v_i = 0$*, there exists scalar functions* $\psi^i_j : \mathbb{R}^d \to \mathbb{R}$*,* $1 \leq i, j \leq k$ *with* $\psi^i_j = -\psi^j_i$ *such that* $v_i = \sum_j \partial_j \psi^i_j$*.*

The Assumption 1 is trivial for $k = 0$. Suppose it is true for $k - 1$ we prove it for $k$: assume $\operatorname{div}_k \mathbf{v} = 0$ and let

$$f_1(x_1, ..., x_n) = \int_0^{x_1} \partial_k v_k(\xi, x_2, ..., x_n) d\xi. \tag{25}$$

Since $\partial_1 f_1 = \partial_x v_k$ one has

$$\partial_1(v_1 + f_1) + \partial_2 v_2 + ... + \partial_{k-1} v_{k-1} = 0.$$

Thanks to Assumption 1 there exists functions $\psi^i_j$ with $\psi^i_j = -\psi^j_i$ such that

$$v_1 + f_1 = \sum_{j=1}^{k-1} \partial_j \psi^1_j, \quad v_i = \sum_{j=1}^{k-1} \partial_j \psi^i_j, \text{ for } 2 \leq i \leq k-1. \tag{26}$$

Now we define

$$f_2(x_1, ..., x_d) = \int_0^{x_1} v_k(\xi, x_2, ..., x_{k-1}, 0, x_{k+1}..., x_d) d\xi - \int_0^{x_k} f_1(x_1, ..., x_{k-1}, \xi, ..., x_d) d\xi, \tag{27}$$

[2] https://math.stackexchange.com/questions/578898

then

$$\partial_k f_2 = -f_1, \tag{28}$$

and using (25) in (27) one gets

$$\partial_1 f_2 = v_k(x_1, ..., x_{k-1}, 0, ..., x_d) - \int_0^{x_k} \partial_k v_k(x_1, ..., x_{k-1}, \xi, ..., x_d) d\xi = -v_k. \tag{29}$$

Now we extend the functions $\psi_j^i$, $1 \leq i, j \leq k-1$ by defining $\psi_k^1 = -\psi_1^k = f_2$ and $\psi_k^i = -\psi_i^k = 0$ for $2 \leq i \leq k$. Then extending the equations (26) with $k$ one has

$$\sum_{j=1}^{k} \partial_j \psi_j^1 = v_1 + f_1 + \partial_k f_2 = v_1, \quad \sum_{j=1}^{k} \partial_j \psi_j^i = v_j, \ 2 \leq i \leq k-1$$

where we used the equality (28) in the first equation. Moreover with the definition of the function $\psi_j^k$ one has

$$\sum_{j=1}^{k} \partial_j \psi_j^k = -\partial_1 f_2 = v_k,$$

thanks to (29). This prove Assumption 1 for $k$. □

### B.4 Proof of Lemma 1

**Lemma.** *Let $\Omega = [-1, 1]^d$ and consider the functions*

$$\psi_j^i(\mathbf{x}) = (x_i^2 - 1)(x_j^2 - 1)\widetilde{\psi_j^i}(\mathbf{x}),$$

*where $\widetilde{\psi_j^i}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ are arbitrary scalar functions satisfying $\widetilde{\psi_j^i} = -\widetilde{\psi_i^j}$. Then the function $\mathbf{v}$ defined in Proposition 2 satisfies $\nabla \cdot \mathbf{v} = 0$ and $\mathbf{v} \cdot \mathbf{n} = 0$ on $\partial\Omega$.*

*Proof.* Indeed since $\psi_i^i = 0$ there is no $\partial_{x_i}$ term which appear in the sum of (7) for the component $v_i$. The term $x_i^2 - 1$ can therefore be factored that is $v_i = 0$ if $x_i = \pm 1$. Hence $\mathbf{v} \cdot \mathbf{n} = 0$ on $\partial\Omega$. □

### B.5 Total derivative expansion

In this subsection we recall the expansion of the Lagrangian derivative $\frac{D}{Dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) = \frac{d}{dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) + (\mathbf{v}(t, \mathbf{x} \cdot \nabla)\mathbf{v}(t, \mathbf{X}(t, \mathbf{x}))$. Assume $\mathbf{X}(t, \mathbf{x})$ follows the ODE (4)

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

We recall that the notation $d/dt$ must be understood as deriving the first variable of $\mathbf{v}(t, \mathbf{X}(t, \mathbf{x}))$ while $D/Dt$ represents the derivative in time of the function $f(t) := \mathbf{v}(t, \mathbf{X}(t, \mathbf{x}))$. By deriving the first and second variable with respect to $t$ one has

$$\frac{D}{Dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) = \frac{d}{dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) + (\frac{d}{dt}\mathbf{X}(t, \mathbf{x}) \cdot \nabla)\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})).$$

Using $\frac{d}{dt}\mathbf{X}(t, \mathbf{x}) = \mathbf{v}(t, \mathbf{X}(t, \mathbf{x}))$ one finally obtains

$$\frac{D}{Dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) = \frac{d}{dt}\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})) + (\mathbf{v}(t, \mathbf{x} \cdot \nabla)\mathbf{v}(t, \mathbf{X}(t, \mathbf{x})).$$

## C Experiment details

We run the experiments on two separate GPUs: a NVIDIA Quadro RTX 8000 and a NVIDIA TITAN X. Our loss is given by the negative log-likelihood (1).

**Toy datasets.** We give the parameters used for the 2D toy experiments in Table 3. All the 2D test cases can be found for example here `https://github.com/rtqichen/ffjord/blob/master/lib/toy_data.py`. We consider a training set of 80K samples and a testing set of 20K samples, 20

| Model | # params (pre-trained model) + GP | epochs | batch size | lr |
|---|---|---|---|---|
| **eight gaussians** | | | | |
| BNAF+GP | (15.4K) + 332 | 2000 | 1000 | $10^{-2}$ |
| **two moons** | | | | |
| BNAF+GP | (15.4K) + 332 | 50 | 1000 | $10^{-2}$ |

**Table 3:** Parameters used for the training of GP flows on the 2D toy examples.

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

**Table 4:** Parameters used for the training of GP flows on the dSprites data set.
* The training is stopped early due to out-of-domain particles.

time steps with a Runge-Kutta 4 discretization and a GP flow with two intermediate layers of size 15. For the Euler penalization we take the exact same parameters with $\lambda = 5 \times 10^{-4}$.

**dSprites datasets.** The VAE architecture used in the experiments is taken from the github repository of Yann Dubois[3]. The parameters used for the GP flows are given in Table 4. For all test cases we consider a GP flow with 15 time steps and three intermediate layers of 50 parameters. 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

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

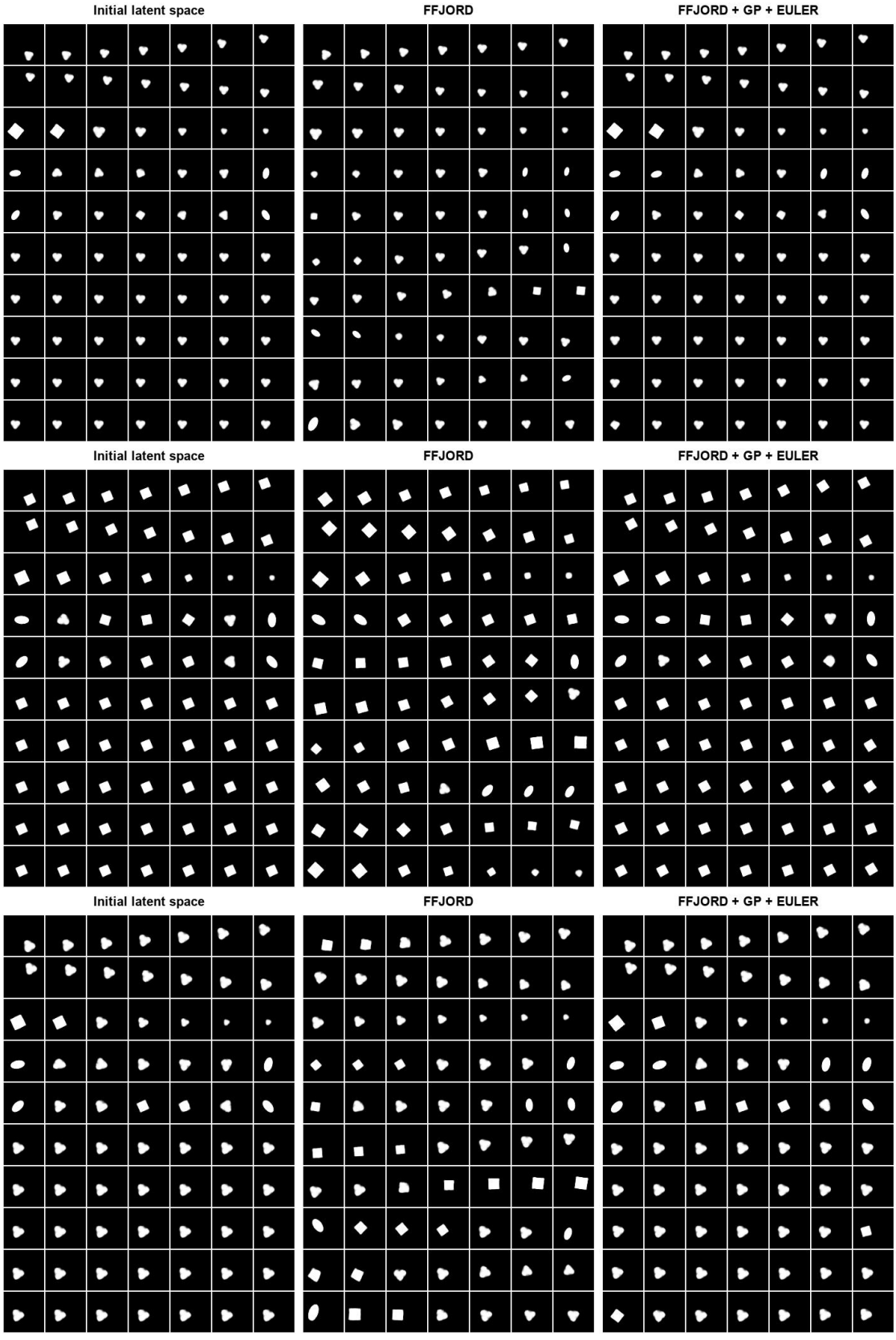


**Figure 5:** Examples of interpolation for the dSprites dataset where each block correspond to the interpolation of a different data point along the 10 dimensions axis represented by the rows. The dimensions are sorted with respect to their KL divergence in the VAE latent space, so the higher rows carry more information while the last rows should leave the image unchanged.

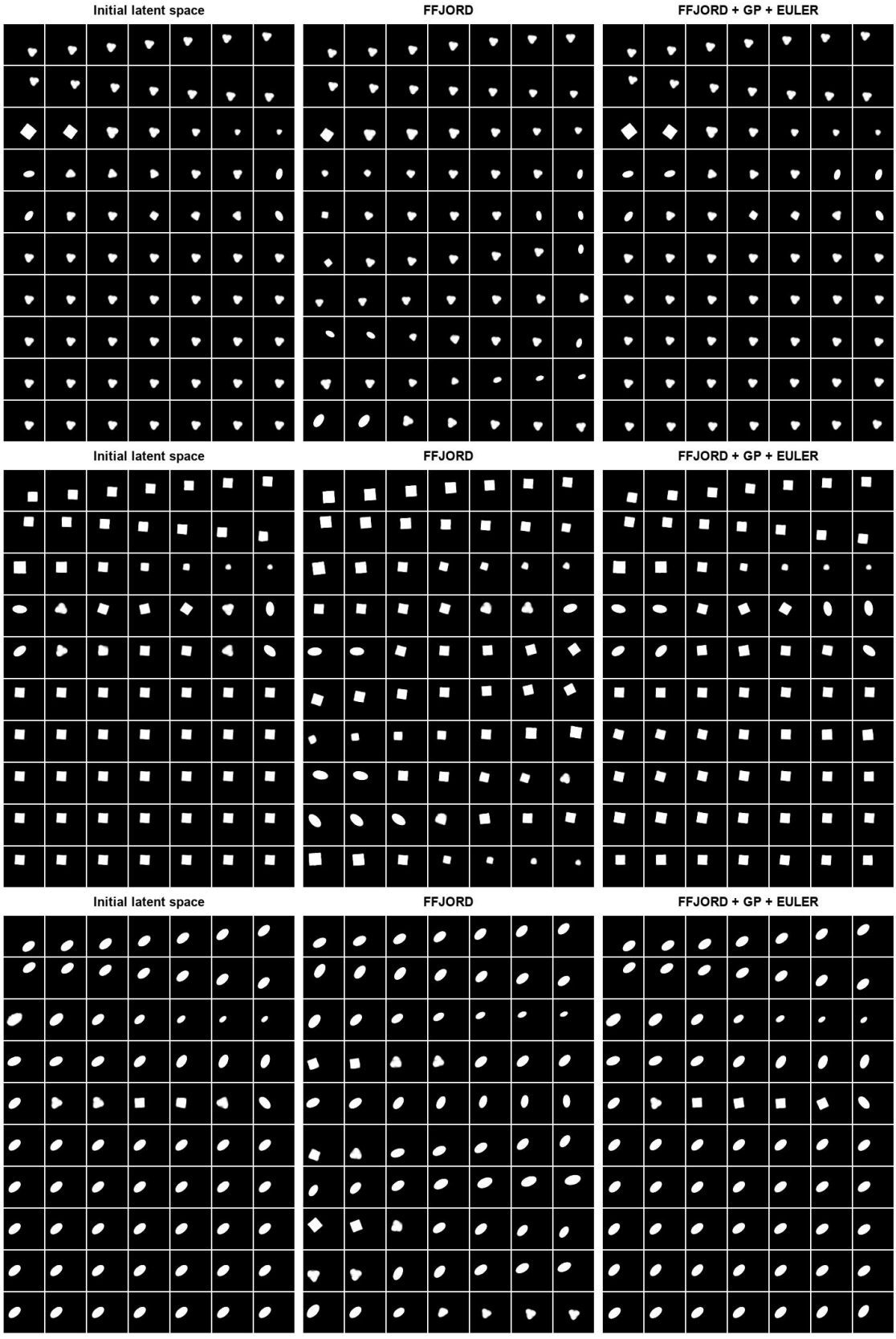


**Figure 6:** Examples of interpolation for the dSprites dataset where each block correspond to the interpolation of a different data point along the 10 dimensions axis represented by the rows. The dimensions are sorted with respect to their KL divergence in the VAE latent space, so the higher rows carry more information while the last rows should leave the image unchanged.