On a Neural Implementation of Brenier's Polar Factorization

Nina Vesseron [1] **Marco Cuturi** [2 1]

Abstract

In 1991, Brenier proved a theorem that generalizes the polar decomposition for square matrices – factored as PSD $\times$ unitary – to any vector field $F : \mathbb{R}^d \to \mathbb{R}^d$. The theorem, known as the polar factorization theorem, states that any field F can be recovered as the composition of the gradient of a convex function u with a measure-preserving map M, namely $F = \nabla u \circ M$. We propose a practical implementation of this far-reaching theoretical result, and explore possible uses within machine learning. The theorem is closely related to optimal transport (OT) theory, and we borrow from recent advances in the field of neural optimal transport to parameterize the potential u as an input convex neural network. The map M can be either evaluated pointwise using u^*, the convex conjugate of u, through the identity $M = \nabla u^* \circ F$, or learned as an auxiliary network. Because M is, in general, not injective, we consider the additional task of estimating the ill-posed inverse map that can approximate the pre-image measure M^{-1} using a stochastic generator. We illustrate possible applications of Brenier's polar factorization to non-convex optimization problems, as well as sampling of densities that are not log-concave.

1. Introduction

Brenier proved, through his seminal polar factorization theorem (1991), that *any* vector field can be decomposed into two simpler elements: Given a reference measure ρ supported on $\Omega \subset \mathbb{R}^d$, For any $F : \Omega \to \mathbb{R}^d$, there exists a convex potential $u : \mathbb{R}^d \to \mathbb{R}$ and a *measure-preserving* map $M : \Omega \to \Omega$ (i.e. on has $M_\sharp \rho = \rho$), such that $F = \nabla u \circ M$. The polar factorization theorem states that *any* vector field, no matter how irregular, can be reshuffled to match that of the gradient of a convex potential, and that this careful reshuffling in space is achieved by the measure-preserving map M. This paper aims to provide a practical approach to recover approximations of the potential u and vector-valued map M using exclusively samples $x_i \sim \rho$ and their associated images $F(x_i)$. We also highlight how a reliable polar factorization solver, coupled with an estimation of a stochastic generator that mimics the measure-valued inverse map M^{-1}, can be used to study the gradient field of non-convex landscapes. We consider, in particular, the case where the field F of interest is the gradient, with respect to the parameters of a neural architecture, of a learning loss. Note that the polar factorization theorem should not be confused with the major theorem from optimal transport closely associated with Brenier, which we recall in §2. That theorem states that the Monge formulation of the optimal transport (OT) problem, which seeks the push-forward map transporting a measure onto another with the least mean displacements (as measured with squared norms) is solved by the gradient of a convex potential.

Existing Implementations. Shortly after (Brenier, 1991), Benamou and Brenier (1994) proposed a numerical approach to decompose a vector field, with an explicit Eulerian (gridded) approach. Lagrangian approaches have been proposed by Gallouët and Mérigot (2018), while Mérigot and Mirebeau (2016) use a semidiscrete OT formulation. Both are used on low-dimensional manifolds as lower level subroutines to solve Euler's equation for incompressible and inviscid fluids (Arnold, 1966). More recently, Morel et al. (2023) proposed to use Brenier's insight to gradually refactor a normalizing flow as the gradient of a convex map, a.k.a a Monge (1781) map, by applying measure-preserving maps for the Gaussian distribution. Their approach does not, however, rely on neural OT solvers, and focuses instead on untangling an existing flow to turn it gradually into the gradient of a convex potential.

Contributions. We propose in this work a neural implementation of the polar factorization theorem that leverages recent advances in neural optimal transport. More precisely,

- After introducing the polar factorization theorem, as well as neural OT solvers, we show how the two blocks of Brenier's result can be recovered using input convex neural networks (ICNN) (Amos et al., 2017). We modify

[1]CREST-ENSAE, IP Paris [2]Apple. Correspondence to: Nina Vesseron <nina.vesseron@ensae.fr>, Marco Cuturi <cuturi@apple.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

the ICNN architecture originally proposed in Amos et al. (2017) and Korotin et al. (2020), to propose quadratic (low-rank + diagonal) positive definite layers at each layer. Starting from an arbitrary field F, we use the modifications proposed by Amos (2023) to (Makkuva et al., 2020b) to train the Brenier convex potential u_θ, that appears in the polar factorization of F.

- We study two alternative parameterizations for the measure-preserving map M: Either implicit, relying on the pointwise evaluation of the convex conjugate of u_θ composed with F, or explicit, through an additional network M_ξ trained to map samples x from ρ to $\nabla u_\theta^* \circ F(x)$.
- Because M is not, in general, injective, we consider the ill-posed problem of inverting M: we approximate a stochastic map I_ψ, parameterized as a generator, that can generate inputs x such as $M(x) = y$ for a given y. We use bridge matching for this task (De Bortoli et al., 2023).
- We use our approach to factorize gradients of surfaces in low dimensions and show how to use our tools to study the critical points of a non-convex energy g. Factorizing $G := \nabla g$ as $\nabla u_\theta \circ M_\xi$, and estimating the stochastic map I_ψ corresponding to M_ξ, our goal is to generate zeros of ∇g. The minimizer of u_θ being $\nabla u_\theta^*(0)$ by definition of convex duality, the points generated as $I_\psi(\nabla u_\theta^*(0), \mathbf{z})$ where $\mathbf{z}$ is a Gaussian noise of suitable size should, in principle, result in points that are roots of ∇g. We use the cross-entropy loss of a small MNIST digits LeNet (LeCun et al., 1998) classifier and show the ability to sample new parameters with low gradient and good performance on the recognition task.

2. Background

This section introduces neural methods that have been proposed to learn Monge maps between two distributions and recalls the polar factorization theorem in its original form.

2.1. Neural Approaches to the Monge Problem

The Monge formulation of the OT problem between two probability measures μ and $\nu \in \mathcal{P}(\mathbb{R}^d)$ seeks a map $T : \mathbb{R}^d \to \mathbb{R}^d$ that transports μ onto ν, while minimizing the following transport cost:

$$\mathcal{W}_2^2(\mu, \nu) := \inf_{\substack{T:\mathbb{R}^d \to \mathbb{R}^d \\ T_\# \mu = \nu}} \int_{\mathbb{R}^d} \tfrac{1}{2}\|x - T(x)\|^2 \mathrm{d}\mu(x) \qquad (1)$$

The existence of an optimal map $T^\star$ is guaranteed under fairly general conditions (Santambrogio, 2015, §1), when e.g. μ has a density w.r.t. the Lebesgue measure. In that case, Brenier's most famous theorem states that the Monge problem (1) has a unique solution, found at the gradient of a convex function $f^\star$ i.e. $T^\star = \nabla f^\star$. That convex function $f^\star$ is itself the solution of the following dual objective:

$$f^\star \in \operatorname*{arg\,inf}_{f \in \mathrm{L}^1(\mu)} \int_{\mathbb{R}^d} f \mathrm{d}\mu + \int_{\mathbb{R}^d} f^* \mathrm{d}\nu \qquad (2)$$

where the f^* is the convex conjugate of f,

$$f^*(y) := \sup_{x \in \mathbb{R}^d} \langle x, y \rangle - f(x)\,. \qquad (3)$$

Note that the star symbol $*$ used for convex-conjugacy should not be confused with the star symbol $\star$, used throughout the paper to denote an optimal solution. The OT map from ν to μ is also given by the inverse of $\nabla f^\star$ when it exists, $\nabla (f^\star)^*$. The goal of neural OT solvers is to estimate $f^\star$ using samples drawn from the source μ and the target distribution ν. Makkuva et al. (2020b); Korotin et al. (2020) have proposed methods that build on input convex neural networks (ICNN), as originally proposed by Amos et al. (2017), to parameterize the potential f as an ICNN. The main difficulty in these methods lies in handling the Legendre transform in (2) of the ICNN variable. To address this difficulty, surrogate networks can be used to replace f^*, and we refer to Amos (2023) for the most recent proposal to refine these implementations using amortized optimization. Neural solvers have been used successfully in various applications, notably in single cell genomics (Bunne et al., 2023; 2022); see also (Huang et al., 2020; Cohen et al., 2021).

2.2. Polar Factorization

Given a probability distribution ρ supported on a bounded set Ω, Brenier's polar factorization theorem states that any vector field $F : \Omega \to \mathbb{R}^d$ can be written as the composition of the gradient of a convex function $\nabla u : \Omega \to \mathbb{R}^d$ with a map $M : \Omega \to \Omega$ that preserves the distribution ρ (ie $M_\# \rho = \rho$). In that decomposition, ∇u is the unique OT map from Brenier's theorem that transports the measure ρ on $F_\# \rho$, since $F_\# \rho = (\nabla u \circ M)_\# \rho = \nabla u_\# (M_\# \rho) = \nabla u_\# \rho$.

Theorem 2.1 (Brenier polar factorization)**.** *Let ρ be a probability measure whose support, $\Omega \subseteq R^d$, is a bounded set and $F : \Omega \to \mathbb{R}^d$ a square-integrable vector field being non degenerate i.e. $\int_{\mathbb{R}^d} \|F\|^2 \mathrm{d}\rho < \infty$ and $\rho(F^{-1}(A)) = 0$ on Lebesgue negligible subsets A of $\mathbb{R}^d$. Then, there exists a convex function $u : \Omega \to \mathbb{R}$ and a map $M : \Omega \to \Omega$ that is measure preserving, i.e. $M_\# \rho = \rho$, such that:*

$$F = \nabla u \circ M\,. \qquad (4)$$

Both M and ∇u are unique.

3. Neural Polar Factorization (NPF)

We describe our method to compute the approximate polar factorization of a field F, using i.i.d samples $(x_1, \ldots, x_n) \sim \rho$ and their evaluations $(F(x_i))_i$. We

first estimate the convex potential u in the decomposition $F = \nabla u \circ M$ using an ICNN u_θ as an OT Brenier (1991) potential using an improved ICNN architecture. Next, we show that the measure-preserving map M can be defined implicitly for any x, by evaluating the convex conjugate of u_θ on $F(x)$: This requires a call to a convex optimization routine at each evaluation. Thanks to our ICNN's strong convexity, the transform (3) is well-posed. To underline the link of that approach to estimate M using u_θ, we use the notation $M_\theta(x)$. Alternatively, we also propose to learn an amortized model for M, by learning a network M_ξ trained on a regression task using paired data samples $\{(x_i, M_\theta(x_i))\}$.

3.1. (Low-Rank + Diagonal) Quadratic Layers in ICNNs

ICNNs provide a neural network parameterization of convex functions. We propose a modification of the original architecture presented in (Amos et al., 2017). Our approach is inspired by the Gaussian initialization outlined in (Bunne et al., 2023) and the low-rank quadratic layers presented in (Korotin et al., 2020). The original ICNN was designed to re-inject the input vector x, transformed by an affine map, at every layer, as can be seen in (Amos et al., 2017, Equation 2, $y \to x$). Korotin et al., §B.2 proposed instead to modify x with multiple low-rank quadratic positive definite (PSD) forms. The PSD constraint ensures convexity of each entry, while the low-rank choice ensures a reasonable number of parameters. We propose quadratic PSD forms that incorporate a *positive* diagonal plus low-rank matrices (Saunderson et al., 2012; Liutkus and Yoshii, 2017):

$$Q_{A,\delta}(x) := \|\delta \circ x\|^2 + \|Ax\|^2 = x^T \left(\mathrm{diag}(\delta) + A^T A\right) x\,.$$

The network has $L+1$ layers for $L \geq 1$; we have highlighted in blue the new PSD (diagonal + low-rank) terms:

$$\begin{aligned} z_0 &= \sigma_0 \left([Q_{A_0^i,\delta_0^i}(x)]_i + B_0 x + c_0 \right), \\ z_{\ell+1} &= \sigma_\ell \left(W_\ell z_\ell + [Q_{A_\ell^i,\delta_\ell^i}(x)]_i + B_\ell x + c_\ell \right), \\ z_{L+1} &= \sigma_L \left({w_L}^T z_\ell + Q_{A_L,\delta_L}(x) + b_L^T x + c_L \right) \in \mathbb{R} \\ u_\theta(x) &= z_{L+1} \end{aligned} \tag{5}$$

In all layers above, the index i spans $1, \ldots, q$, where q is the size of the state vectors $z_\ell \in \mathbb{R}^q$. This augmented ICNN is parameterized with the following family of parameters,

$$\begin{aligned} \theta = \big(& W_{1:L-1} \in (\mathbb{R}_+^{q\times q})^L, w_L \in \mathbb{R}_+^q, \\ & \left(\delta_{0:L-1}^i \in (\mathbb{R}_+^d)^L, A_{0:L-1}^i \in (\mathbb{R}^{r\times d})^L\right)_{i=1\ldots q}, \\ & \delta_L \in \mathbb{R}_+^d, A_L \in \mathbb{R}^{r\times d}, \\ & B_{0:L-1} \in (\mathbb{R}^{q\times d})^L, b_L \in \mathbb{R}^d, \\ & c_{0:L-1} \in (\mathbb{R}^q)^L, c_L \in \mathbb{R} \big)\,. \end{aligned} \tag{6}$$

The activation functions σ_ℓ are convex, non-decreasing nonlinear and all parameters in red in addition to all diagonal vectors δ must be non-negative to ensure convexity.

3.2. Estimating the Convex Potential u

Starting from the existence result outlined in (4), we recover, by applying the push-foward map F on ρ, that

$$F_\sharp \rho = (\nabla u \circ M)_\sharp \rho = \nabla u_\sharp (M_\sharp \rho) = \nabla u_\sharp \rho\,.$$

Since u is a convex function, it optimally transports ρ on $\nabla u_\# \rho$ in the Monge sense. Therefore, the defining feature of u is that ∇u is the Monge map from ρ to $F_\sharp \rho$. We use Amos' solver (2023) to estimate the potential u that pushes ρ onto $F_\# \rho$, from the empirical measures $\rho_n := \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ and $F_\# \rho_n = \frac{1}{n}\sum_{i=1}^n \delta_{F(x_i)}$. Using this solver consists of parameterizing the convex function u as an ICNN u_θ following §3.1 and parameterizing ∇u^* directly by an auxiliary vector-valued network V_ϕ. The auxiliary network V_ϕ is learned by minimizing the objective:

$$\mathcal{L}_{\text{convex-dual}}(\phi) = \frac{1}{n}\sum_{i=1}^n \|V_\phi(F(x_i)) - \nabla u_\theta^*(F(x_i))\|^2$$

One can show, using Danskin's envelope theorem (1966), that $\nabla u_\theta^*(y)$ is the maximizer of the convex conjugate (3) problem for u at y,

$$\nabla u_\theta^*(y) = \arg\sup_x \langle x, y\rangle - u_\theta(x)$$

Because u_θ is strictly convex, we compute the optimal solution solving $u^*(F(x))$ with a conjugate solver, e.g. gradient ascent, (L)BFGS (Liu and Nocedal, 1989) or ADAM (Kingma and Ba, 2014). We call a conjugate solver CS any algorithm that, for a given pair (u, y), outputs an approximation of $\nabla u^*(y)$. This results in the loss:

$$\mathcal{L}_{\text{convex-dual}}(\phi) = \frac{1}{n}\sum_{i=1}^n \|V_\phi(F(x_i)) - \mathrm{CS}(u_\theta, F(x_i))\|^2 \tag{7}$$

In practice, the latter is initialized with the predictions of V_ϕ, which considerably reduces the number of iterations required for the solver to converge when V_ϕ starts making correct predictions. The parameters of the network u_θ are then updated alternatively, by taking steps along the gradients of the original dual objective of Makkuva et al. (2020b):

$$\begin{aligned} \mathcal{L}_{\text{Monge}}(\theta) = \frac{1}{n}\sum_{i=1}^n & u_\theta(x_i) + \langle V_\phi(F(x_i)), F(x_i)\rangle \\ & - u_\theta(V_\phi(F(x_i))) \end{aligned} \tag{8}$$

3.3. Estimating the Measure-Preserving Map M

In the polar decomposition of F, ∇u is tasked with transporting ρ on $F_\# \rho$, the measure-preserving map M ensures

then that $F = \nabla u \circ M$. To express M as a function of F and u, one simply has to apply the inverse of ∇u on both sides. When u is strictly convex, we simply rely on the identity $\nabla u^* \circ \nabla u = \mathrm{Id}$ to obtain:

$$M = \nabla u^* \circ F \tag{9}$$

Evaluating M using a Conjugate Solver. Given a conjugate solver CS and the estimate u_θ for the ground truth potential u, we can inject them in (9) to get an estimation $\mathrm{CS}(u_\theta, F(x))$ of $M(x)$ for a given x. Since this estimation depends on θ, we define that approximation as $M_\theta(x)$,

$$M_\theta(x) := \mathrm{CS}(u_\theta, F(x))\,, \tag{10}$$

with a slight abuse of notation, since θ should not be understood as a parameter parameterizing M, but instead defining it implicitly through u_θ and CS.

Neural Estimation for M. While M_θ does indeed provide an estimate of M, it may be convenient to parameterize the measure-preserving map of interest as a neural network M_ξ, defined with an independent set of parameters ξ. Borrowing a page from amortized optimization (Amos et al., 2023), M_ξ can be used to initialize the conjugate solver used to estimate M_θ or even replace it when M_ξ is sufficiently accurate. Furthermore, the parameterization of M by M_ξ is sometimes necessary when, e.g., F is only given on a few samples, and one wishes to evaluate M at any point. The neural map M_ξ is then trained to minimize the following mean-squared error:

$$\mathcal{L}_{\text{preserving}}(\xi) = \frac{1}{n}\sum_{i=1}^{n} \|M_\xi(x_i) - \mathrm{CS}(u_\theta, F(x_i))\|^2\,. \tag{11}$$

Note that while the loss in (11) resembles (7), the network V_ϕ takes the transported point $F(x_i)$ as an input, whereas M_ξ is only given x_i.

Evaluating The Measure Preservation of M. In both cases, M_θ, as evaluated with a conjugate solver, or its independently evaluated neural counterpart M_ξ should be measure-preserving. Indeed, we will use (as in Figure 1, bottom center plots) any departure from the identity

$$M_\# \rho = (\nabla u^* \circ F)_\# \rho = (\nabla u^*)_\# (F_\# \rho) = \rho\,,$$

as a way to assess the quality of our factorization.

3.4. Sampling according to the pre-image measure M_θ^{-1}

Measure-preserving maps M are not invertible in general (Ryff, 1970), a well-known example in 1D being the doubling map defined as $M(x) = 2x \bmod 1$ that preserves the Lebesgue measure rescaled to the interval $[0, 1]$. This non-invertibility is of particular interest in the optimization and sampling applications we propose. For a given y, our goal will therefore be to generate inputs x such that $M_\theta(x) = y$. To this end, we learn a generative process to sample according to the posterior density

$$\pi_\theta(x|y) = \frac{\mathbf{1}_{y=M_\theta(x)}\rho(x)}{\int_x \mathbf{1}_{y=M_\theta(x)}\rho(x)\mathrm{d}x}. \tag{12}$$

We rely on the augmented bridge matching procedure presented in De Bortoli et al. (2023) to learn the drift of the stochastic differential equation (SDE) formulated in (13) so that, on input $X_0 = y$, the generated samples X_1 be distributed according to $\pi_\theta(x|y)$ (12).

$$\mathrm{d}X_t = (X_\psi(X_0, X_t) - X_t)/(1-t)\mathrm{d}t + \sigma \mathrm{d}B_t \tag{13}$$

De Bortoli et al.'s approach refines the bridge matching procedures that have been recently used to solve inverse problems (Somnath et al., 2023; Liu et al., 2023; Chung et al., 2024), by augmenting the learnable part of the drift X_ψ with the initial point X_0 of the SDE. This slight adjustment allows to correctly recover the coupling measure $(M_\theta, \mathrm{Id})_\# \rho$ from the paired samples $\{(x_i, F(x_i))\}_{i=1}^n$ when X_ψ is parameterized using a multilayer perceptron trained according to Algorithm 1.

Algorithm 1 Training of X_ψ

$u_\theta \longleftarrow$ Trained ICNN s.t. $\nabla u_{\theta\#}\rho \approx F_\#\rho$
Initialize X_ψ
while not converged **do**
  Draw a sample $(x_i, F(x_i))$
  Compute $y_i = \mathrm{CS}(u_\theta, F(x_i))$
  Sample $t \sim \mathcal{U}([0,1])$
  Sample $z_i \sim \mathcal{N}(0, I_d)$
  $x_t := (1-t)y_i + t x_i + \sigma(t(1-t))^{1/2} z_i$
  $\mathcal{L}_\psi \leftarrow \frac{1}{n}\|X_\psi(y_i, x_t) - x_i\|^2$
  Update X_ψ using $\nabla \mathcal{L}_\psi$
end while

The optimized network X_ψ is then plugged in (13) that we solve with Heun's method as implemented in `diffrax` (Kidger, 2021) using S discretization steps. Given a sample y from $M_{\theta\#}\rho$, solving the SDE (13) using $X_0 = y$ allows to generate an output X_1 distributed according to the posterior density (12). To alleviate the notations, we call I_ψ the generative process such that $I_\psi(y, \mathbf{z})$ is the output X_1 returned by the differential equation solver associated to (13) on the input $X_0 = y$ when the injected gaussian noise $\mathbf{z}$ has been drawn from $\mathcal{N}(0, I_d)^{\otimes S}$

$$I_\psi(y, \mathbf{z}) = \mathrm{SDE}(X_\psi, y, \mathbf{z}),\ y \in \mathbb{R}^d,\ \mathbf{z} \in \mathbb{R}^{d \times S}.$$

To generate several inputs x from $\pi_\theta(x|y)$, one only needs to inject different noises $\mathbf{z} \sim \mathcal{N}(0, I_d)^{\otimes S}$ in $I_\psi(y, \cdot)$, i.e.

$$I_\psi(y, \cdot)_\# \mathcal{N}(0, I_d)^{\otimes S} \approx \pi_\theta(\cdot|y).$$

4. NPF to Study Non-Convex Potentials g

In this section, we focus on the polar factorisation of the gradient field ∇g, where g is a non-convex function of interest. We show how computing the NPF of ∇g together with the inverse map I_ψ can be used to explore the space of critical points of g.

4.1. On using the Inverse Map I_ψ of ∇g

NPF on $G = \nabla g$. Let $g : \mathbb{R}^d \to \mathbb{R}$ be a function of interest supported on a bounded set $\Omega \subset \mathbb{R}^d$. Assuming that $\nabla g : \mathbb{R}^d \to \mathbb{R}^d$ meets the requirements of (4), Brenier's polar factorization states the existence of a convex function u and a measure-preserving M that preserves the rescaled Lebesgue measure on Ω, $\mathcal{L}_\Omega$ such that:

$$\nabla g = \nabla u \circ M.$$

For a given vector v, the points in Ω whose gradient with respect to g is equal to v are all transported by M on the same point $\nabla u^*(v)$ i.e.

$$M\left(\{x \in \Omega : \nabla g(x) = v\}\right) = \{\nabla u^*(v)\}.$$

In particular, the critical points of g are all mapped by M onto the minimizer of the function u, which is $\nabla u^*(0)$.

On Extracting the Critical Points of g. When the NPF of ∇g is learned, resulting in u_θ, M_ξ and I_ψ, composing I_ψ with $\nabla u_\theta *$ provides an inversion process for ∇g. Generating an input point x_v whose gradient is v can in fact be done by first sampling $\mathbf{z} \sim \mathcal{N}(0, I_d)^{\otimes N}$ and successively applying $\nabla u_\theta *$ and I_ψ to v:

$$x_v = I_\psi(\nabla u_\theta^*(v), \mathbf{z}), \text{ where } \mathbf{z} \sim \mathcal{N}(0, I_d)^{\otimes S}.$$

As a special case, sampling the critical points of g is done by taking $v = 0$ in the above procedure with different noises $\mathbf{z}$. Note, however, that this convexification requires estimating the polar factorization of ∇g as well as the inverse map I_ψ over the entire space Ω, which is computationally expensive. To optimize the g function, we propose instead to combine this method with the Langevin Monte Carlo (LMC) algorithm to correctly estimate the polar factorization of ∇g around the minimums of g.

4.2. A LMC Method Assisted by NPF.

LMC algorithm Given a smooth log-concave density

$$\pi(x) = \frac{e^{-g(x)}}{\int_{x \in \mathbb{R}^d} e^{-g(x)} \mathrm{d}x}, \tag{14}$$

with $g : \mathbb{R}^d \longrightarrow \mathbb{R}$, the Langevin Monte Carlo algorithm can sample from π by starting from $x^{(0)}$ to iterate

$$x^{(k+1)} = x^{(k)} - \gamma \nabla g(x^{(k)}) + \sqrt{2\gamma} z^{(k)}, \quad z^{(k)} \sim \mathcal{N}(0, I_d).$$

When g is non-convex, the LMC algorithm lacks guarantees (Roberts and Tweedie, 1996; Cheng and Bartlett, 2018; Dalalyan and Karagulyan, 2019). In particular, when g has multiple local minima, the generated samples are highly correlated as the particles originating from the LMC algorithm often get stuck in some basins. For this reason, the LMC algorithm has been combined with methods enabling global jumps between modes (Pompe et al., 2020; Gabrié et al., 2022) to sample multi-modal distributions.

Sampling with Known Polar Factorization for ∇g. In this paragraph, we assume that the polar factorization $(\nabla u, M)$ and stochastic inverse map M^{-1} of ∇g are known. To sample the modes of $\pi(x) \propto e^{-g(x)}$ when g is non-convex, one can run the LMC algorithm on the convex function u and sample back using an inverse generator M^{-1}:

$$\begin{aligned} y^{(k)} &= M(x^{(k)}) \\ y^{(k+1)} &= y^{(k)} - \gamma \nabla u(y^{(k)}) + \sqrt{2\gamma} z^{(k)} \\ x^{(k+1)} &= M^{-1}(y^{(k+1)}, z^{(k+\frac{1}{2})}). \end{aligned}$$

The LMC step on u allows to move along a new descent direction or exploration direction to reach y_{k+1} while M^{-1} randomly generates a point $x^{(k+1)} \in \Omega$ whose gradient for g is $\nabla u(y_{k+1})$. This way, the neighborhoods of g's critical points are uniformly sampled, and a particle does not get stuck in one minimum as M^{-1} permits global moves between all the basins. Because it is difficult to differentiate a minimum from a saddle point or a maximum when sampling critical points using the polar factorization of ∇u, this procedure should be combined with Langevin steps on g to escape non-minimum critical points. The following paragraph details the sampling algorithm and complements it by showing how the polar factorization of ∇u can be learned while sampling.

Unknown Polar Factorization for ∇g When the polar factorization is unknown, we propose an algorithm that learns the polar factorization of ∇g as well as the inverse map I_ψ using the generated particle trajectories. The algorithm alternates between N Langevin steps on g and N Langevin steps on u_θ, while M_θ and I_ψ allow to transition

between the two spaces. Algorithm 2 details the steps of the procedure. The notation LMC $(u_\theta, \gamma, y_i^{(k)}, N)$ means that N LMC steps are performed on the function u_θ with a time step of γ starting from the point $y_i^{(k)}$.

Algorithm 2 LMC-NPF

Initialize u_θ and I_ψ
Initialize the particles $\{x_i^{(0)}\}_{1\leq i\leq n}$
$k \leftarrow 0$
while $k < k_{max}$ **do**
  if $k \mod N = 0$ **then**
    $y_i^{(k)} = M_\theta(x_i^{(k)})$
    $y_i^{(k+1)} = \text{LMC}\ (u_\theta, \gamma, y_i^{(k)}, N)$
    $x_i^{(k+1)} = I_\psi(y_i^{(k+1)}, \mathbf{z})$ with $\mathbf{z} \sim \mathcal{N}(0, I_d)^{\otimes S}$
  else
    $x_i^{(k+1)} = x_i^{(k)} - \gamma \nabla g(x_i^{(k)}) + \sqrt{2\gamma} z_i^{(k)}$
  end if
  Update u_θ, I_ψ with $\{(x_i^{(k)}, \nabla g(x_i^{(k)}))\}_{1\leq i\leq n}$
  $k \leftarrow k+1$
end while

The main insight of the proposed sampling algorithm is that LMC steps permit the exploration of the space locally, while NPF provides and stores a more global viewpoint, that is able to propose moves to potentially worthy areas.

5. Experiments

5.1. Accuracy Metrics for NPF

Assess NPF's Accuracy. When a field G is only available through samples, the following three criteria, evaluated on unseen samples (or test set) $\{(x_j, G(x_j))\}_{1\leq j\leq m}$, are used to assess whether the estimated polar factorization is correct.

- To measure that the distributions $\nabla u_{\theta\#}\rho$ and $G_\#\rho$ are close, we compute the Sinkhorn divergence S_ε (Ramdas et al., 2017; Genevay et al., 2018; Peyré et al., 2019) between the two point clouds $(G(x_j))_{1\leq j\leq m}$ and $(\nabla u_\theta(x_j))_{1\leq j\leq m}$. To quantify the scale of that measurement, we compare it with the distance between two batches of fixed size drawn from $(G(x_j))_{1\leq j\leq m}$. We also visualize this proximity by embedding the two point clouds using the TSNE algorithm (Van der Maaten and Hinton, 2008) and superimpose them.
- The second criterion assesses whether M_ξ is measure-preserving. Similarly, this is numerically estimated by computing the Sinkhorn divergence between the empirical measures associated with $(x_j)_{1\leq j\leq m}$ and $(M_\xi(x_j))_{1\leq j\leq m}$, and visualized with a TSNE embedding.
- Finally, we evaluate the L_2 distance between G and $\nabla u_\theta \circ M_\xi$ using the test set. Note that when M_θ is used (rather than M_ξ), that criterion is not useful since it only assesses the quality of the conjugate solver.

Assess the Generative Inverse Map I_ψ. Given y, we should be able to sample among the antecedents of y by M_θ using the multivalued map I_ψ. To quantify that, we estimate the average distance between the probability associated to the density $\pi_\theta(x|y)$ (12) and $I_\psi(y,\cdot)_\#\mathcal{N}(0,I_d)^{\otimes S}$ from samples. Given the finite test set $\{(x_j, M_\theta(x_j))\}_{1\leq j\leq m}$, it is unlikely to find a multitude of points with the same image. For this reason, we approximate $M_\theta^{-1}(M_\theta(x_k))$, by constructing the set

$$\mathcal{B}_\alpha(x_k) = \{x_j : \|M_\theta(x_j) - M_\theta(x_k)\|_2 \leq \alpha\}$$

and choose α such that the cardinal of $\mathcal{B}_\alpha(x_k)$ is 128. We then compute the sinkhorn divergence between the predictions of I_ψ on $M_\theta(\mathcal{B}_{\alpha_k}(x_k))$ and $\mathcal{B}_{\alpha_k}(x_k)$ and average it over all the x_k. We compare the obtained value with the distance of two batches of fixed size drawn independently from the $(x_j)_{1\leq j\leq m}$. We also evaluate the fact that the stochastic map $I_\psi \circ \nabla u_\theta^*$ approximates G^{-1} by computing the quantity, using MC samples for $\mathbf{z}$,

$$\frac{1}{m}\sum_{j=1}^{m} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,I_d)^{\otimes S}} \text{cosine}(G\circ I_\psi(\nabla u_\theta^*(G(x_j)), \mathbf{z}), G(x_j)) \tag{15}$$

which relies on the cosine similarity metric defined as $\text{cosine}(x,y) = \frac{\langle x,y\rangle}{\|x\|\|y\|}$.

Sinkhorn Divergence Parameter ε The ε parameter used to compute sinkhorn divergence must be adapted to the scale of the data under consideration. In all our experiments, ε is set to $\varepsilon = 0.05\ \mathbb{E}_{x,x'\sim\rho}\|x - x'\|^2$ when computing divergence in the source space (ie to assess the accuracy of M_ξ or I_ψ). Similarly, ε is set to $\varepsilon = 0.05\ \mathbb{E}_{x,x'\sim\rho}\|G(x) - G(x')\|^2$ when computing divergence in the target space (ie to assess the accuracy of u_θ). In practice, the expectations are approximated using 2048×2048 MC samples.

5.2. NPF of Topographical Data

Dataset. We use the Python package `elevation` to get the elevation of three regions of the world: Chamonix, London, and Cyprus. We estimate the gradients associated with the elevation in these regions with finite-differences, and obtain three datasets composed of (latitude, longitude) points paired with their gradients. We learn the polar factorization $(\nabla u_\theta, M_\xi)$ of the underlying gradient field as well as the inverse map I_ψ. Because in these examples, G is only given through samples, it can be interesting to parameterize the measure-preserving map using a neural network M_ξ. To assess the quality of our method NPF, we used a 85% training / 15% test split. More details can be found in the appendix.

	Chamonix
$S_\varepsilon(\nabla u_{\theta\#}\rho_{2048}, G_\#\rho_{2048})$	0.36
$S_\varepsilon(G_\#\rho_{2048}, G_\#\rho'_{2048})$	0.31
$S_\varepsilon(M_{\xi\#}\rho_{2048}, \rho_{2048})$	0.0022
$S_\varepsilon(\rho_{2048}, \rho'_{2048})$	0.0015
$\mathbb{E}_{x\sim\rho_n}[\|\|G(x) - \nabla u_\theta \circ M_\xi(x)\|_2]$	0.96
$\mathbb{E}_{x,\mathbf{z}}[S_\varepsilon((I_\psi(M_\theta(\mathcal{B}_\alpha(x)), \mathbf{z}), \mathcal{B}_\alpha(x))]$	0.046
$S_\varepsilon(\rho_\ell, \rho'_\ell)$	0.039

Table 1. Polar factorization and Inverse multivalued map metrics for learning the gradient of the elevation in Chamonix area. For these metrics, ρ_n and ρ'_n are two empirical measures created from $n = 2048$ samples drawn independently from the test set. Likewise, ρ_ℓ and ρ'_ℓ are two empirical measures created from $\ell = 128$ samples.

Polar Factorization Results. Table 1 shows that the estimated NPF is accurate: the Sinkhorn divergence between the predicted distribution $\nabla u_{\theta\#}\rho_n$ and the target distribution $G_\#\rho_n$ is of the same magnitude as the divergence between two batches ρ_n, ρ'_n of the same size taken from the target. Similarly, the Sinkhorn divergence between ρ_n and its image by M_ξ is of the same order as the distance between two batches of same size drawn from the source. The reconstruction of G is also quite satisfactory as corroborated visually (Figure 1).

Inverse Map Results. The data from Table 1 indicates that I_ψ generates the antecedents of the images by M_θ accurately: the estimated quantity $\mathbb{E}_{x,\mathbf{z}}[S_\varepsilon((I_\psi(M_\theta(\mathcal{B}_\alpha(x)), \mathbf{z}), \mathcal{B}_\alpha(x))]$ which is approximated using MC samples, is comparable to the distance between two batches of size 128 drawn from the source distribution. To visualize these performances, we transported the samples $(G(x_j))_{1\leq j\leq m}$, that store gradients of the elevation, using $I_\psi \circ \nabla u^*_\theta$ that should estimate the inverse generative map G^{-1}. We expect very high gradients to be sent to points where the elevation varies rapidly, such as the sides of mountains in the Chamonix example. To visualize where a gradient was sent, we plot a point at this localization and color it according to the norm of the gradient from which it originates. We compare the image generated by this process with the one obtained by coloring directly the points $(x_j)_{1\leq j\leq m}$ using their associated gradients. In the three cases (Chamonix, London, Cyprus), the two images look quite similar (Figure 2), showing the quality of our reconstruction.

5.3. Learn an NN Optimization Landscape using NPF

In this experiment, we consider a minimal neural architecture capable of classifying MNIST digits. Inspired by the LeNet architecture (LeCun et al., 1998), we use two convolutional layers, each followed by a Relu and a max pooling operation. A classification layer leads to an output layer of 10 neurons, followed by a softmax. The loss function is the cross entropy, computed with MNIST train dataset minibatches of size 128, and the vector field under study is the gradient of that loss for the $d = 222$ parameters of the neural network. The loss landscape of a non-linear neural network being very chaotic (Li et al., 2018), we do not expect to learn the polar factorization of the associated gradient field perfectly over the all optimization space Ω. The optimization space we are considering is $\Omega = [-1, 1]^{222}$.

Polar Factorization and Inverse Map Results. According to Table 2, we see that, overall, NPF manages to learn that vector field, but it lacks, as expected, accuracy in some parts of the space. This is, e.g., revealed visually using the TSNE plot from Figure 3. Similarly, I_ψ can be used to invert M_θ according to Table 2 and the histogram associated with the cosine similarity on Figure 3 confirms that we can choose a certain gradient v and use $I_\psi \circ \nabla u^*_\theta$ to generate classifier weights whose gradient is approximately v. In particular, I_ψ allows the generation of correct critical points (Figure 12), which, however, have a low accuracy. This is due to our uniform sampling procedure in the space $[-1, 1]^{222}$ that only reveals bad critical points with an accuracy of 10% which is the performance of a classifier with random weights as shown in Figure 12.

$S_\varepsilon(\nabla u_{\theta\#}\rho_n, G_\#\rho_n)$	97.4 ± 6.5
$S_\varepsilon(G_\#\rho_n, G_\#\rho'_n)$	93.6 ± 6.2
$\mathbb{E}_{k,\mathbf{z}}[S_\varepsilon((I_\psi(M_\theta(\mathcal{B}_\alpha(x_k)), \mathbf{z}), \mathcal{B}_\alpha(x_k))]$	107.3 ± 0.7
$S_\varepsilon(\rho_\ell, \rho'_\ell)$	105.7 ± 0.5
$\mathbb{E}_{y\sim G_\#\rho_n,\mathbf{z}}\text{cosine}(G \circ I_\psi(\nabla u^*_\theta(y), \mathbf{z}), y)$	0.81 ± 0.20

Table 2. Polar factorization and Inverse multivalued map metrics for learning the gradient of the MNIST classifier loss function. The metrics are computed using emprical distributions of respectively $n = 2048$ and $\ell = 128$ samples.

5.4. Learning NPF using Gradient Flow of Particles

In §5.3, we requested that NPF learns the entire gradient field. This of course limits the ability, given a certain budget of samples, to provide a good approximation of critical points through the inverse generative map. In particular, our uniform sampling procedure does not allow to reveal interesting critical points. In this experiment, we train NPF using gradient descent trajectories to focus on those areas. To do this, we initialize 1024 particles randomly and have them follow a gradient flow. We use these trajectories' samples to learn NPF and sample critical points of the classifier.

Polar Factorization and Inverse Map Results. We observe that NPF is faithful near good accuracy basins: the Sinkhorn divergence between the distribution generated by ∇u_θ and the target is of the same order as that between

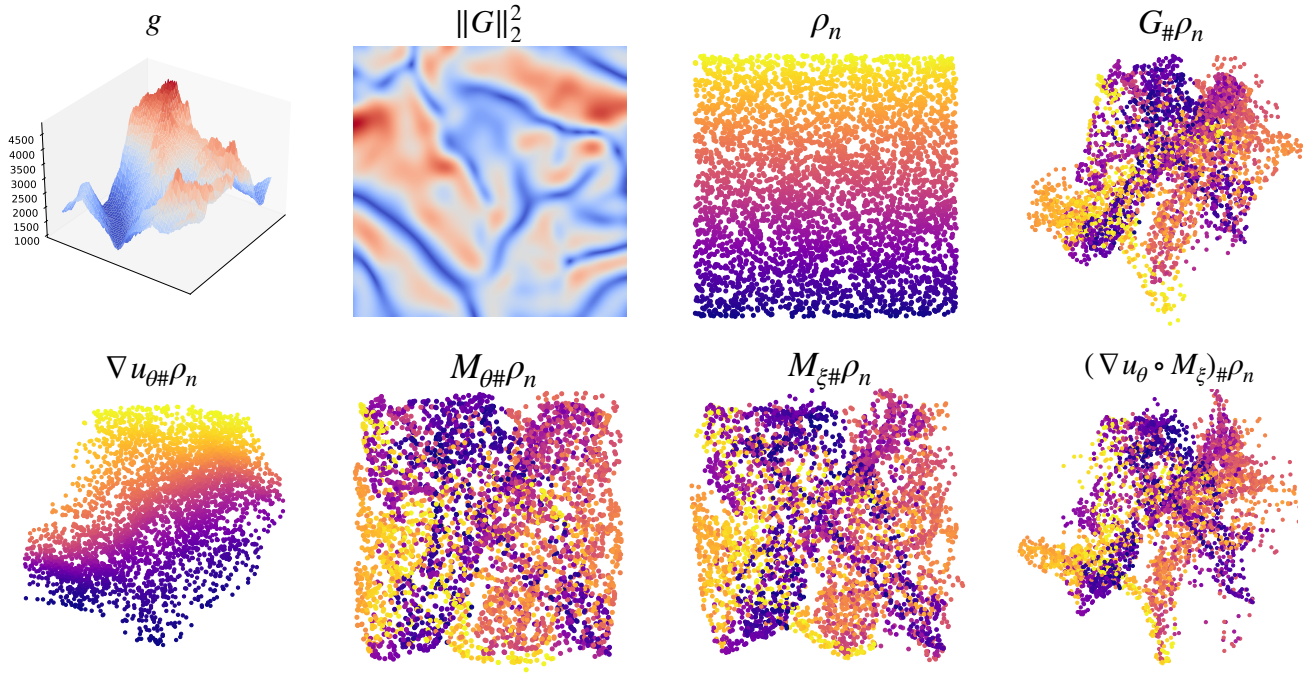

Figure 1. The g function under study is the elevation in the Chamonix area (France). The figures show the respective action of the vector fields involved in the polar factorization of ∇g on a sample measure ρ_n. We observe that $\nabla u_{\theta\#}\rho_n \approx G_\#\rho_n$. Both implicit and explicit measure-preserving maps M_θ (10) as well as the explicit network M_ξ trained with the loss (11) permutes the points of the distribution, ensuring that $G \approx \nabla u_\theta \circ M_\xi$ while $(M_\xi)_\#\rho_n \approx (M_\theta)_\#\rho_n \approx \rho_n$.

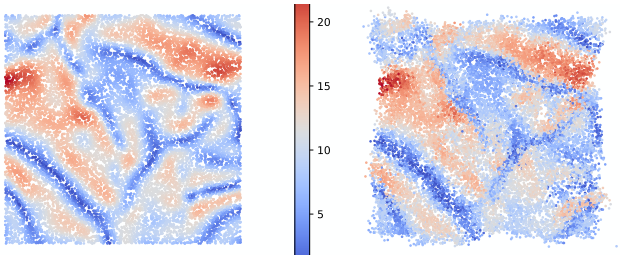

Figure 2. I_ψ's ability to replace gradients in the original Ω space for the example of Chamonix region's elevation gradient. The figure on the right is generated by returning the gradients $\nabla g_\#\rho_n$ to their initial position in the image via $I_\psi \circ \nabla u_\theta^*$. This position is then colored according to the initial gradient norm (before transport). We can compare the result with the image on the left, generated by sampling uniformly in Ω space and colored according to the norm of their gradient.

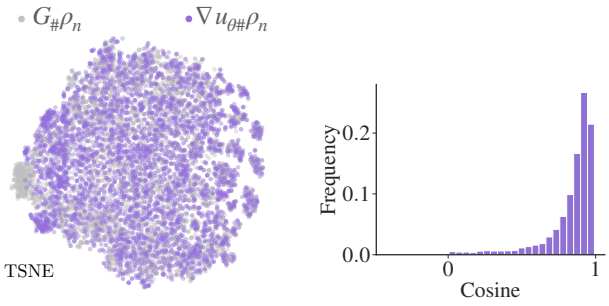

Figure 3. Performance of the NPF and the inverse map I_ψ on the §5.3 MNIST classifier experiment where the loss gradient is learned over the entire space Ω. The TSNE allows to visualize in 2D the overlap of the predicted distribution $\nabla u_{\theta\#}\rho_n$ with the target distribution $G_\#\rho_n$ while the cosine similarity, mentioned in §5.1, shows that I_ψ permits to accurately generate weights associated with a given gradient.

two batches of size 2048. As for I_ψ, the gradients of the generated weights do have a norm close to 0, and the cosine similarity distribution reveals that the direction of the gradients is globally learned (Figure 13). Moreover, we can see in Figure 14 that the critical weights generated contain a large part of valid minima.

5.5. LMC-NPF on MNIST

We use LMC-NPF (Algorithm 2) to sample the loss of the MNIST classifier considered in §5.3. Our sampling algorithm is preceded by a warm-up containing particle descents to explore good minima before sampling.

Sampling Algorithm Results. Following the warm-up, the TSNE (Figure 5) as well as Figure 4 show that our sampling algorithm proposes high-accuracy weights that are completely different from the minima found during the warm-up period. Since LMC-NPF alternates between Langevin steps on g and Langevin steps on u_θ, we demonstrate that these minima had indeed been discovered through the use of NPF by running a LMC algorithm initialized with the final warm-up particles. The latter was parameterized the same way as the one used in LMC-NPF, with the same number of iterations. We observe that the LMC algorithm samples around the warm-up particles but does not detach itself from them. This confirms that the use of PFNet in the sampling procedure permits the discovery of new local minima.

5.6. ICNNs Benchmark

We compare three different ICNN architectures: the one proposed in Amos (2023), the same architecture but with an extra full quadratic layer at the end and ours (with a rank of 1 for the intermediate quadratic layers) making sure that the architectures have a comparable number of parameters. In the first experiment, the starting measure is a d-dimensional standard Gaussian while the target is a gaussian of mean 0 and covariance matrix $\mathrm{diag}(1, 2, ..., d)$. The $\mathcal{L}_2$ unexplained variance percentage (Korotin et al., 2021) from Table 4 indicates that the architecture we propose provides a better estimation of the OT map for all the dimensions we considered. In the second experiment, the starting measure is still a d-dimensional standard gaussian and the objective is to map a mixture of gaussians whose modes have various sizes. The estimated Sinkhorn divergences between the generated distribution and the target (Table 5) shows that our proposed architecture is the one that best fits the multimodal distribution.

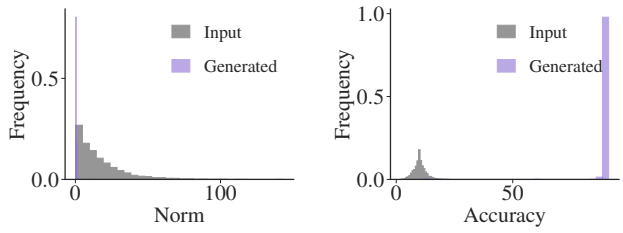

Figure 4. Characteristics of the points sampled using LMC-NPF for the §5.5 MNIST classifier experiment. In gray the classifier weights have been drawn uniformly in the Ω optimization space, while in purple the weights have been sampled using Algorithm 2. Generated samples are critical points which are good minima, as shown by the accuracy statistics.

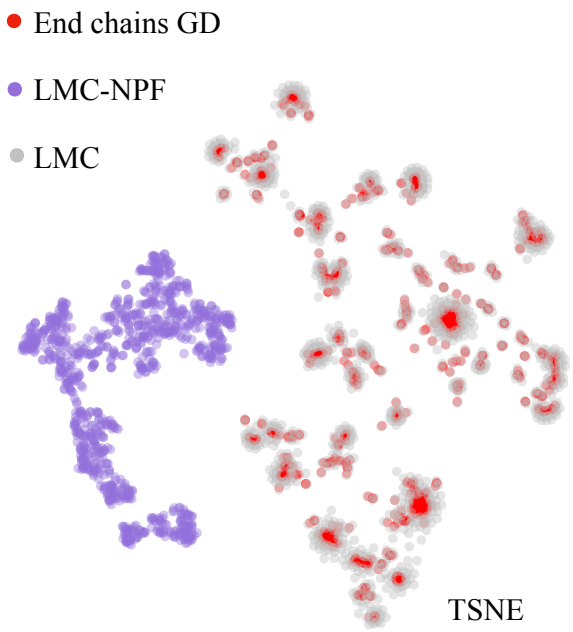

Figure 5. Results of LMC-NPF applied to the MNIST classifier loss function. The TSNE is used to represent the final particles resulting from descent trajectories during the Warm-up period (in red), the particles sampled by the LMC algorithm (in grey), and the particles sampled by LMC-NPF (in purple).

6. Conclusion

Brenier's polar factorization is arguably one of the most far-reaching results discovered in analysis in the last century, underpinning the better known Brenier theorem on the existence of solutions to the Monge (1781) problem. We proposed in this work the first implementation, to the best of our knowledge, of that factorization that is applicable to higher-dimensional settings. To do so, we have used the recently proposed machinery of neural optimal transport solvers. Beyond simply exploiting this result, we have also proposed to estimate a multivalued map that approximates the inverse of the measure-preserving map component in the polar factorization. We have shown that such an inverse map can be of potential use to sample the optimization landscape of non-convex potentials. An interesting direction for perfecting the sampling algorithm would be to reweight the samples according to their probability, in the same vein as SMC samplers (Del Moral et al., 2006). This would require knowledge of the probability distribution generated by the generative model I_ψ, which is not possible with the current methodology.

Acknowledgements

This work was performed using HPC resources from GENCI–IDRIS (Grant 2023-103245).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Brandon Amos. On amortizing convex conjugates for optimal transport. In *The Eleventh International Conference on Learning Representations*, 2023.

Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

Brandon Amos et al. Tutorial on amortized optimization. *Foundations and Trends® in Machine Learning*, 16(5): 592–732, 2023.

Vladimir Arnold. Sur la géométrie différentielle des groupes de lie de dimension infinie et ses applications à l'hydrodynamique des fluides parfaits. In *Annales de l'institut Fourier*, volume 16, pages 319–361, 1966.

JD Benamou and Y Brenier. A domain decomposition method for the polar factorization of vector fields. *Contemporary Mathematics*, 157:231–231, 1994.

Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

Charlotte Bunne, Andreas Krause, and Marco Cuturi. Supervised training of conditional monge maps. *Advances in Neural Information Processing Systems*, 35:6859–6872, 2022.

Charlotte Bunne, Stefan G Stark, Gabriele Gut, Jacobo Sarabia Del Castillo, Mitch Levesque, Kjong-Van Lehmann, Lucas Pelkmans, Andreas Krause, and Gunnar Rätsch. Learning single-cell perturbation responses using neural optimal transport. *Nature Methods*, 20(11):1759–1768, 2023.

Xiang Cheng and Peter Bartlett. Convergence of langevin mcmc in kl-divergence. In *Algorithmic Learning Theory*, pages 186–211. PMLR, 2018.

Hyungjin Chung, Jeongsol Kim, and Jong Chul Ye. Direct diffusion bridge using data consistency for inverse problems. *Advances in Neural Information Processing Systems*, 36, 2024.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Samuel Cohen, Brandon Amos, and Yaron Lipman. Riemannian convex potential maps. In *International Conference on Machine Learning*, pages 2028–2038. PMLR, 2021.

Arnak S Dalalyan and Avetik Karagulyan. User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311, 2019.

John M Danskin. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664, 1966.

Valentin De Bortoli, Guan-Horng Liu, Tianrong Chen, Evangelos A Theodorou, and Weilie Nie. Augmented bridge matching. *arXiv preprint arXiv:2311.06978*, 2023.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

Marylou Gabrié, Grant M Rotskoff, and Eric Vanden-Eijnden. Adaptive monte carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, 119(10), 2022.

Thomas O Gallouët and Quentin Mérigot. A lagrangian scheme à la brenier for the incompressible euler equations. *Foundations of Computational Mathematics*, 18(4):835–865, 2018.

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018.

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.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

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

Alexander Korotin, Vage Egiazarian, Arip Asadulaev, Alexander Safin, and Evgeny Burnaev. Wasserstein-2 generative networks. In *International Conference on Learning Representations*, 2020.

Alexander Korotin, Lingxiao Li, Aude Genevay, Justin M Solomon, Alexander Filippov, and Evgeny Burnaev. Do neural optimal transport solvers work? a continuous wasserstein-2 benchmark. *Advances in neural information processing systems*, 34:14593–14605, 2021.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Guan-Horng Liu, Arash Vahdat, De-An Huang, Evangelos A Theodorou, Weili Nie, and Anima Anandkumar. I^2sb: Image-to-image schrödinger bridge. *arXiv preprint arXiv:2302.05872*, 2023.

Antoine Liutkus and Kazuyoshi Yoshii. A diagonal plus low-rank covariance model for computationally efficient source separation. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2017.

Ashok Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR, 2020a.

Ashok Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason Lee. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR, 2020b.

Quentin Mérigot and Jean-Marie Mirebeau. Minimal geodesics along volume-preserving maps, through semidiscrete optimal transport. *SIAM Journal on Numerical Analysis*, 54(6):3465–3492, 2016.

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences*, pages 666–704, 1781.

Guillaume Morel, Lucas Drumetz, Simon Benaïchouche, Nicolas Courty, and François Rousseau. Turning normalizing flows into monge maps with geodesic gaussian preserving flows. *Transactions on Machine Learning Research*, 2023.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Emilia Pompe, Chris Holmes, and Krzysztof Łatuszyński. A framework for adaptive mcmc targeting multimodal distributions. *The Annals of Statistics*, 48(5):2930 – 2952, 2020.

Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47, 2017.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363, 1996.

John V Ryff. Measure preserving transformations and rearrangements. *Journal of Mathematical Analysis and Applications*, 31(2):449–458, 1970.

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

J. Saunderson, V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky. Diagonal and low-rank matrix decompositions, correlation matrices, and ellipsoid fitting. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1395–1416, 2012.

Vignesh Ram Somnath, Matteo Pariset, Ya-Ping Hsieh, Maria Rodriguez Martinez, Andreas Krause, and Charlotte Bunne. Aligned diffusion schrödinger bridges. In *The 39th Conference on Uncertainty in Artificial Intelligence*, 2023.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008.

A. Computation of the convex conjugate

Given a function convex function u, and a point y, the objective $J_u(x, y) = \langle y, x\rangle - u(x)$ is concave with respect to x and $u^*(y) = \sup_x J_u(x, y)$ can be computed using optimization algorithms like gradient ascent, (L)BFGS or Adam. As for $\nabla u^*(y)$, taking the gradient for y necessitates differentiate through a supremum. In our case, u is strictly convex a.e. and the supremum becomes a maximum :

$$u^*(y) = \max_x J_u(x, y)$$

Danskin's envelope theorem (1966) allows to differentiate through this maximum and to write:

$$\begin{aligned}\nabla_y u^*(y) &= \nabla_y \max_x J_u(x, y)\\ &= (\nabla_y J_u(x, y))(x^*(y))\end{aligned}$$

where $x^*(y)$ is the optimal x that maximizes $J_u(x, y)$. Because $\nabla_y J_u(x, y) = x$, we get that

$$\nabla_y u^*(y) = x^*(y)$$

B. Preconditionned LMC

$$x^{(k+1)} = x^{(k)} - \gamma \nabla f(x^{(k)}) + \sqrt{2\gamma} z^{(k)}, \quad z^{(k)} \sim \mathcal{N}(0, I_d)$$

By replacing ∇f with its polar factorization, the procedure becomes:

$$x^{(k+1)} = x^{(k)} - \gamma \nabla u \circ M(x^{(k)}) + \sqrt{2\gamma} z^{(k)}$$

By studying $y^{(k)} = M(x^{(k)})$, one can see that the LMC procedure implies doing a preconditioned LMC algorithm on the convex function u.

$$\begin{aligned}y^{(k+1)} &= M(x^{(k+1)})\\ &= M\left(x^{(k)} - \gamma \nabla u(y^{(k)}) + \sqrt{2\gamma} z^{(k)}\right)\\ &= M(x^{(k)}) + J_M(x^{(k)})\left[-\gamma \nabla u(y^{(k)}) + \sqrt{2\gamma} z^{(k)}\right]\\ &\quad + \circ(\|\varepsilon\|)\\ &= y^{(k)} - \gamma J_M(x^{(k)}) \nabla u(y^{(k)}) + \sqrt{2\gamma} J_M(x^{(k)}) z^{(k)}\\ &\quad + \circ(\|\varepsilon\|)\end{aligned}$$

with $\varepsilon = -\gamma \nabla u(y^{(k)}) + \sqrt{2\gamma} z^{(k)}$. One can note that the preconditioned matrix $H = J_M(x^{(k)})$ is not necessarily positive definite.

C. Augmented Bridge Matching

Given a coupling $\Pi_{0,1}$ and random variables (X_0, X_1), the augmented bridge matching algorithm (De Bortoli et al., 2023) aims at learning a stochastic dynamic mapping between X_0 and X_1 that preserves the coupling $\Pi_{0,1}$.

In the probability space of path measures $\mathcal{P}(\mathcal{C}([0, 1], \mathbb{R}^d))$, let $\mathcal{M}$ denotes the path measures associated to the SDE $dX_t = v_t(X_t)\mathrm{d}t + \sigma_t \mathrm{d}B_t$, the functions σ and v being locally Lipschitz. Given a path measure $\mathbb{Q} \in \mathcal{M}$, the diffusion bridge of $\mathbb{Q}$ which is the distribution of $\mathbb{Q}$ conditioned on both endpoint is denoted by $\mathbb{Q}_{|0,1}$. The set of path measures considered to bridge $\mathcal{P}(X_0)$ and $\mathcal{P}(X_1)$ according to the coupling $\Pi_{0,1}$ is $\Pi_{0,1}\mathbb{Q}_{|0,1} = \int_{\mathbb{R}^d \times \mathbb{R}^d} \mathbb{Q}_{|0,1}(.|x_0, x_1)\Pi_{0,1}(\mathrm{d}x_0, \mathrm{d}x_1)$. In De Bortoli et al. (2023), the authors showed that under mild conditions, $\Pi_{0,1}\mathbb{Q}_{|0,1}$ was associated to the following SDE:

$$\mathrm{d}X_t = \{b_t(X_t) + \sigma_t^2 u_t\}\mathrm{d}t + \sigma_t \mathrm{d}B_t, \qquad X_0 \sim \mu$$

with $u_t = \mathbb{E}_{\mathbb{P}_{1|0,t}}\left[\nabla \log \mathbb{Q}_{1|t}(X_1|X_t)|X_0, X_t\right]$ where $\mathbb{Q}_{1|t}$ and $\mathbb{P}_{1|0,t}$ are respectively the conditional distribution of $\mathbb{Q}$ at time 1 given the state at time t and the conditional distribution of $\mathbb{P}$ at time 1 given the coupling state at time 0 and t.

This SDE gives a way to sample from $\Pi_{0,1}$ by first sampling $X_0 \sim \mu$ and then discretize the SDE to get X_1. Because u_t is intractable, it is approximated by a neural network u_t^θ learned to minimize the regression loss:

$$\int_0^1 \lambda_t \mathbb{E}[\|u_t^\theta(X_0, X_t) - \nabla \log \mathbb{Q}_{1|t}(X_1|X_t)\|^2] \mathrm{d}\mathbb{P}(X_0, X_t, X_1)$$

A particular case of diffusion bridge is the Brownian bridge $\mathbb{Q}_{|0,1}$ for which $v = 0$ and $\sigma_t = \sigma$ which is the one usually used in practice.

D. ICNNs Benchmark

We compare three different architectures: Linear which is the one proposed in Amos (2023), FQuad with the same architecture but with a extra full quadratic layer at the end and ours (with a rank of 1 for the intermediate quadratic layers). Given input dimension d, all ICNNs have 4 layers, with hidden states z_i of size $[d, d, d, d]$ for our method, $[2d, 2d, 2d, 2d]$ for Linear, FQuad so that architectures have a comparable number of parameters. The ICNN optimization is done with a constant learning rate of 0.0005 for $d = 32$ and $d = 128$. The networks have been trained for 60000 iterations. In all experiments, the distances have been computed with batches of size 4096. In the first experiment, the starting measure μ is a d-dimensional standard Gaussian, the target ν is a gaussian of mean 0 and covariance matrix $\mathrm{diag}(1, 2, ..., d)$. Because both source and target distribution are gaussians, the OT map is known in closed form Peyré et al. (2019) and the $\mathcal{L}_2$ unexplained variance percentage (Makkuva et al., 2020a; Korotin et al., 2021) is used to assess the accuracy of the computed OT map. The estimation of this metric is done using batches of size 2048. In the second experiment, the starting measure μ is still a d-dimensional standard Gaussian and the objective is to map multimodal data whose modes have various sizes. More precisely, the target distribution is composed of 7 gaussians whose means are respectively : $(30, 0, 30, 0, 30, 0, ...)$ $(-30, 0, -30, 0, -30, 0, ...)$ $(0, -30, 0, -30, 0, -30, ...)$ $(\frac{30}{\sqrt{2}}, \frac{30}{\sqrt{2}}, \frac{30}{\sqrt{2}}, \frac{30}{\sqrt{2}}, ...)$ $(\frac{30}{\sqrt{2}}, \frac{-30}{\sqrt{2}}, \frac{30}{\sqrt{2}}, \frac{-30}{\sqrt{2}}, ...)$ $(\frac{-30}{\sqrt{2}}, \frac{30}{\sqrt{2}}, \frac{-30}{\sqrt{2}}, \frac{30}{\sqrt{2}}, ...)$ and their covariance matrices are : I_d, $2I_d$, $3I_d$, $4I_d$, $5I_d$, $6I_d$, $7I_d$. This time, the OT map is not known in closed form and we only use the Sinkhorn divergence to estimate the distance between the generated distribution $\nabla u_{\theta\#}\mu$ and the target ν in order to compare the three architectures.

dimension d	32	128
Linear	20 577	328 065
FQuad	20 673	328 449
Ours	15 714	247 170

Table 3. Number of parameters for the 3 architectures under consideration.

dimension d	32	128
Linear	0.72 ± 0.05	1.26 ± 0.08
FQuad	0.38 ± 0.03	0.97 ± 0.06
Ours	$\mathbf{0.031 \pm 0.003}$	$\mathbf{0.082 \pm 0.002}$

Table 4. $\mathcal{L}_2$ unexplained variance percentage for the 3 architectures considered when the target is the multivariate gaussian.

dimension d	32	128
Linear	548 ± 75	$246 \times 1e^1 \pm 42 \times 1e^1$
FQuad	452 ± 77	$176 \times 1e^1 \pm 14 \times 1e^1$
Ours	$\mathbf{234 \pm 42}$	$\mathbf{104 \times 1e^1 \pm 25 \times 1e^1}$

Table 5. Sinkhorn divergence results for the 3 architectures considered when the target is the mixture of gaussians.

E. Topography Experiments

E.1. Creation of the dataset

We used the Python package `elevation` to get the elevation of three different regions of the globe: Chamonix, London, and Cyprus. Given the latitudes and longitudes of the desired area, `elevation` returns a grid of the area with the elevation

value at each grid point. For the Chamonix example, we obtained 323932 points $(x, y) \in \mathbb{R}^2$ and their corresponding elevation. We dequantized the elevations by adding a uniform noise on [0,1] to them before using a Gaussian filter to make the gradients smoother. To do this, we used the function `gaussian_filter` from the `scipy` library. We then numerically estimate the gradients associated with the elevation and obtain a dataset of 323932 points in $\mathbb{R}^2$ and the associated gradients in $\mathbb{R}^2$ for the example of Chamonix. We obtained data for the Cyprus and London regions in the same way.

E.2. London and Cyprus results

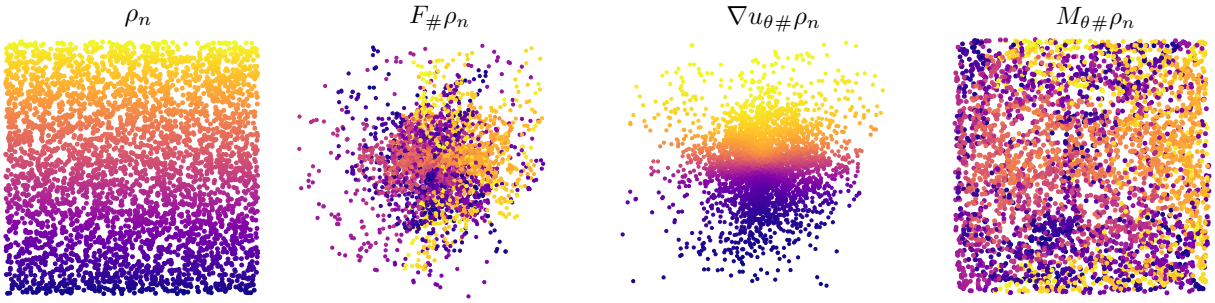

Figure 6. Respective actions of the learned vector fields associated with the polar factorization of the elevation gradient in the London area.

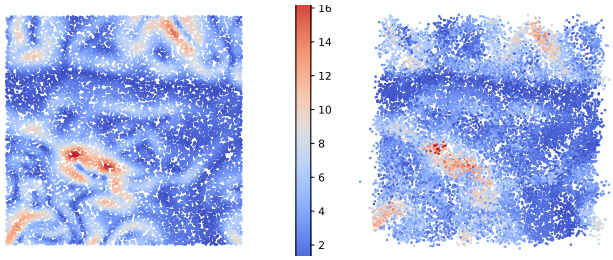

Figure 7. I_ψ's ability to replace gradients in the original Ω space for the example of London region's elevation gradient.

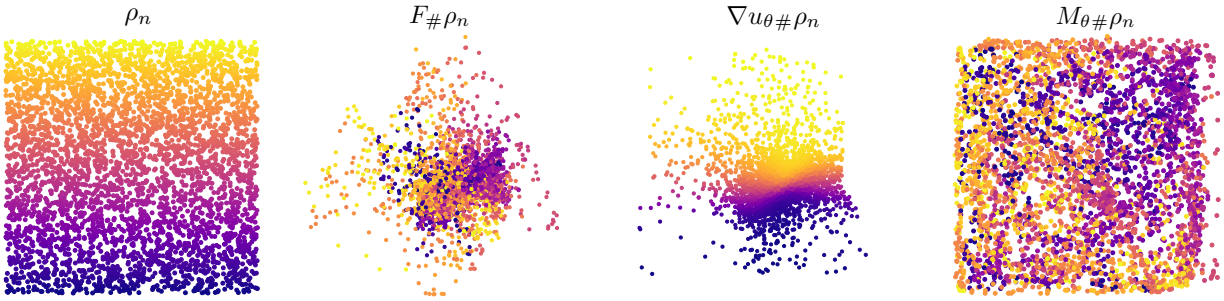

Figure 8. Respective actions of the learned vector fields associated with the polar factorization of the elevation gradient in the Cyprus neighborhood.

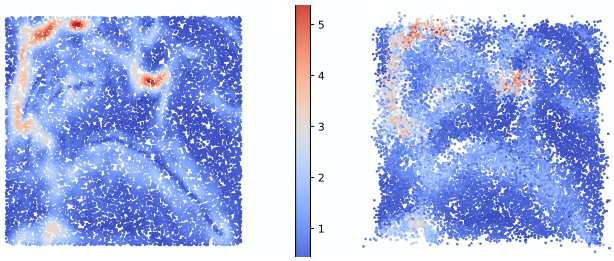

Figure 9. I_ψ's ability to replace gradients in the original Ω space for the example of Cyprus region's elevation gradient.

	Chamonix	London	Chypre
$W_\varepsilon(\nabla u_{\theta\#}\rho_n, F_\#\rho_n)$	0.36	0.26	0.021
$W_\varepsilon(\nabla F_\#\rho_n, F_\#\rho'_n)$	0.31	0.25	0.020
$\mathbb{E}_{k,\mathbf{z}}\left[S_\varepsilon((I_\psi(M_\theta(\mathcal{B}_\alpha(x_k)),\mathbf{z}),\mathcal{B}_\alpha(x_k))\right]$	0.046	0.054	0.038
$S_\varepsilon(\rho_{128}, \rho'_{128})$	0.039	0.035	0.021

Figure 10. NPF and I_ψ performances for the topography experiments, $n = 2048$.

F. LeNet classifier Experiments

F.1. LeNet classifier architecture

The LeNet classifier architecture used for the experiments is composed of two convolutive layers followed by a relu activation function and a max pooling; it ends with a dense layer as described in Figure 11. The optimization space that we consider is $\Omega = [-1, 1]^{222}$.

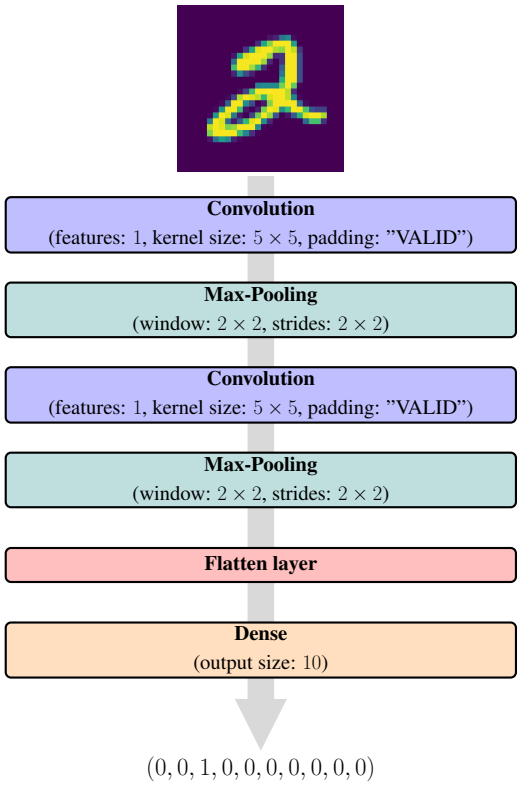

Figure 11. Le Net classifier architecture used in experiments.

F.2. Complementary graphs for the MNIST classifier experiments

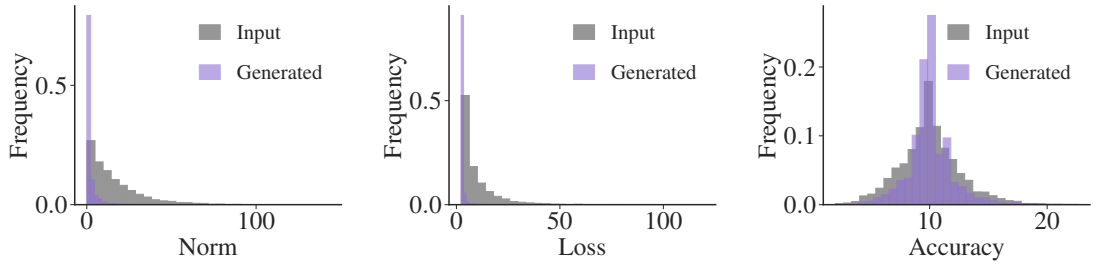

Figure 12. Characteristics of the generated critical points for the §5.3 MNIST classifier experiment. In gray the classifier weights have been drawn uniformly in the Ω optimization space, while in purple the weights have been drawn using $I_\psi(\nabla u^*_\theta(0), \mathbf{z})$.

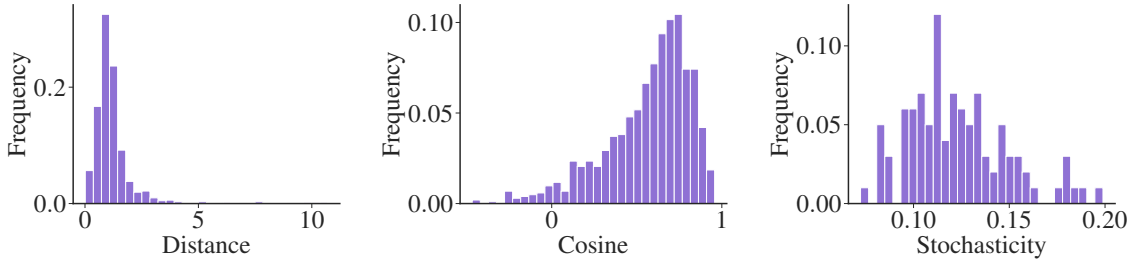

Figure 13. Performances of I_ψ for the §5.4 MNIST classifier experiment and stochasticity of the MNIST loss. The distance and cosine plots demonstrate the ability of I_ψ to correctly generate weights with a fixed gradient. For the stochasticity plot, the classifier weights are fixed to the weights obtained after gradient descent and different minibatches of MNIST images are used to compute the gradient of the loss function. The stochasticity plot shows the distribution of the sinkhorn divergence between two gradient batches computed from the same weights.

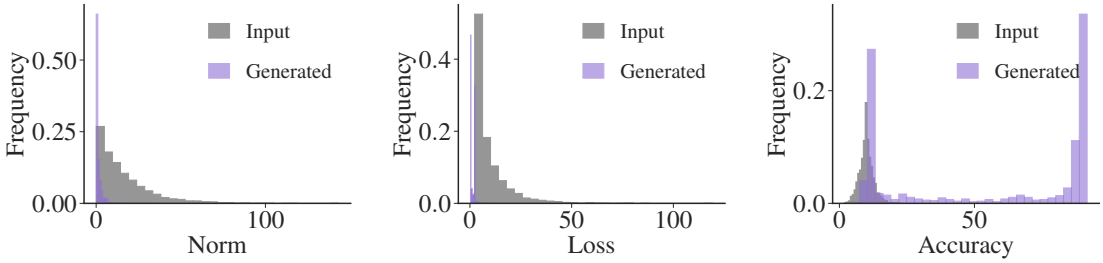

Figure 14. Characteristics of the generated critical points for the §5.4 MNIST classifier experiment. In gray the classifier weights have been drawn uniformly in the Ω optimization space, while in purple the weights have been drawn using $I_\psi(\nabla u^*_\theta(0), \mathbf{z})$.

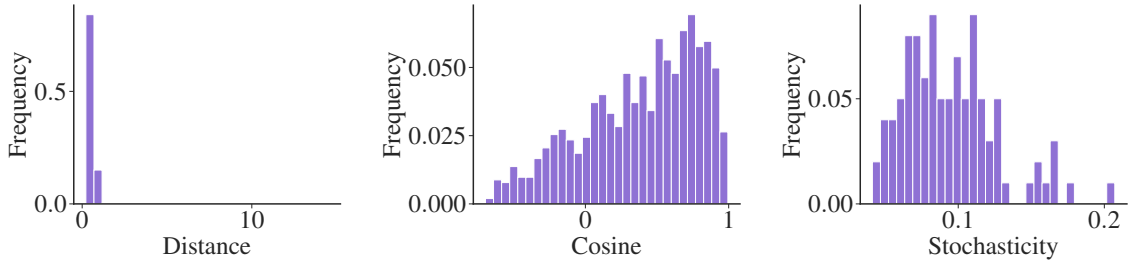

Figure 15. Performances of I_ψ for the §5.5 sampling MNIST classifier experiment. The distance and cosine plots demonstrate the ability of I_ψ to correctly generate weights with a fixed gradient. For the stochasticity plot, the classifier weights are fixed to the final sampled particles and different minibatches of MNIST images are used to compute the gradient of the loss function. The stochasticity plot shows the distribution of the sinkhorn divergence between two gradient batches computed from the same weights.

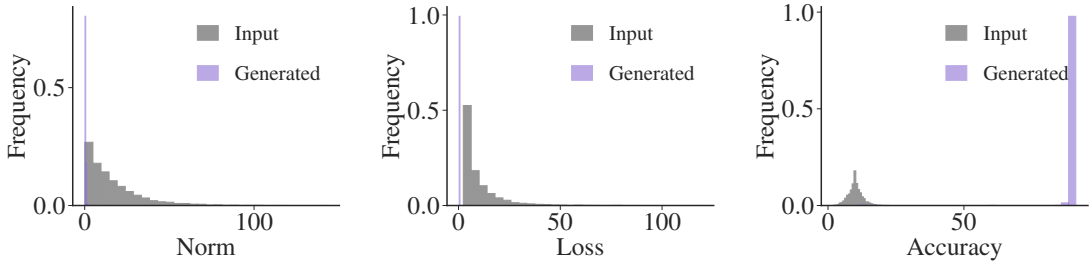

Figure 16. Characteristics of the points sampled using Algorithm 2 for the §5.5 MNIST classifier experiment. In gray the classifier weights have been drawn uniformly in the Ω optimization space, while in purple the weights have been sampled using Algorithm 2. The generated samples are critical points which are good minima, as shown by the accuracy statistics.

G. Hyperparameters

G.1. Parameterize u_θ

In all experiments, the convex function u is parameterized using an ICNN u_θ whose architecture is detailed in §3.1. The rank of the quadratic term $Q_{A,\delta}(x)$ is always taken equal to 1, which means that A is a row matrix. **We noted that it was necessary to choose smooth activation functions in u's parameterization to avoid convergence problems with the conjugate solvers, that occur especially in high dimension**. This is why we have favored the use of ELU (Clevert et al., 2015) activations in the u parameterization rather than Relu activations.

G.2. Computation of u_θ^*

The use of a conjugate solver is necessary to compute the loss functions of V_ϕ, M_ξ, I_ψ and to estimate M_θ. In all cases, the objective is to estimate the gradient of u_θ's conjugate at a given point y: $\nabla(u_\theta)^*(y)$. For a given experiment, that justifies the use of the same conjugate solver parameters for these different applications. We relied on ADAM solver for the computation of the convex conjugate as it runs faster than LBFGS on our examples and use Amos implementation. The two hyperparameters that remain to be set are the maximum number of iterations given to the solver to converge and the tolerance factor at which the norm of the gradient is considered small enough for the solver to have converged. These two hyperparameters are strongly dependent on the dimension of the problem as well as on the function u_θ and, therefore, on the distributions ρ and $F_\#\rho$. To amortize the number of iterations required for the solver to converge, it is always initialized with the prediction of the V_ϕ network that is trained in conjunction with u_θ.

G.3. Parameterize V_ϕ

We use an MLP with 2 hidden layers of size 512 and Relu activation functions to parameterize V_ϕ in all our experiments.

model	hyperparameter	value
u_θ	activation function	elu
	architecture	[64, 64, 64, 64]
	b1	0.50
	b2	0.50
	scheduler	cosine decay
	initial learning rate	0.001
	α	0.10
	scheduler steps	50000
	steps	50000
I_ψ	activation function	silu
	architecture	[256, 256, 256]
	scheduler	cosine decay
	initial learning rate	0.001
	α	0.010
	scheduler steps	50000
	steps	50000
	σ	0.1
V_ϕ	activation function	relu
	architecture	[512, 512]
	b1	0.9
	b2	0.999
	scheduler	cosine decay
	initial learning rate	0.0005
	α	0.010
	scheduler steps	50000
	steps	50000
conjugate solver	name	Adam
	max iteration	200
	gtol	0.001

Figure 17. Hyperparameters used for the topography experiments (the same hyperparameters have been used for Chamonix, London, and Cyprus).

G.4. Parameterize M_ξ

The measure-preserving map M is parameterized by a neural network only when the vector field under study is available through samples only. This is the case in the topography examples where M_ξ is parameterized by an MLP with 2 hidden layers of size 512 and Relu activation functions.

G.5. Parameterize X_ψ

The learned part of the drift X_ψ is parameterized using an MLP, and we use Silu activation functions which is the classic choice for parameterizing the drift X_ψ. The same hyperparameters have been used for the Chamonix, London, and Cyprus cases.

model	hyperparameter	value
u_θ	activation function	elu
	architecture	[128, 128, 128, 128]
	b1	0.5
	b2	0.5
	scheduler	cosine decay
	initial learning rate	0.001
	α	0.01
	scheduler steps	10000
	steps	10000
I_ψ	activation function	silu
	architecture	[512, 512]
	scheduler	cosine decay
	initial learning rate	0.0005
	α	0.01
	scheduler steps	50000
	steps	50000
	σ	1.0
V_ϕ	activation function	relu
	architecture	[512, 512]
	b1	0.9
	b2	0.999
	scheduler	cosine decay
	initial learning rate	0.0005
	α	0.01
	scheduler steps	10000
	steps	10000
conjugate solver	name	Adam
	max iterations	700
	gtol	0.1

Figure 18. Hyperparameters used for experiment 6.3.

model	hyperparameter	value
u_θ	activation function	elu
	architecture	[128, 128, 128, 128]
	b1	0.5
	b2	0.5
	scheduler	cosine decay
	initial learning rate	0.0001
	α	0.1
	scheduler steps	30000
I_ψ	activation function	silu
	architecture	[512, 512]
	scheduler	constant learning rate
	learning rate	0.0005
	σ	0.1
V_ϕ	activation function	relu
	architecture	[512, 512]
	b1	0.9
	b2	0.999
	scheduler	constant learning rate
	learning rate	0.0005
conjugate solver	name	Adam
	max iteration	1000
	gtol	0.001
particles	steps	60000
	particules	1024
	γ (step size for the gradient descent)	0.1

Figure 19. Hyperparameters used for experiment 6.4.

model	hyperparameter	value
u_θ	activation function	elu
	architecture	[128, 128, 128, 128]
	b1	0.5
	b2	0.5
	scheduler	cosine decay
	initial learning rate	0.0001
	α	0.1
	scheduler steps	30000
I_ψ	activation function	silu
	architecture	[512, 512]
	scheduler	constant learning rate
	learning rate	0.0005
	σ	0.1
V_ϕ	activation function	relu
	architecture	[512, 512]
	b1	0.9
	b2	0.999
	scheduler	constant learning rate
	learning rate	0.0005
conjugate solver	name	Adam
	max iteration	1000
	gtol	0.001
particles	steps	60000
	LMC multiplicative coefficient in front of ∇f	1000
	LMC multiplicative coefficient in front of ∇u	1000
	particules	1024
	warming steps	30000
	N	200
	τ_f (step size for LMC steps on f)	0.0001
	τ_u (step size for LMC steps on u)	0.0001

Figure 20. Hyperparameters used for experiment 6.5.