arXiv:2106.00774v2 [stat.ML] 14 Jun 2021

Optimizing Functionals on the Space of Probabilities with Input Convex Neural Networks

David Alvarez-Melis
Microsoft Research
daalvare@microsoft.com

Yair Schiff
IBM Watson
yair.schiff@ibm.com

Youssef Mroueh
IBM Research AI
mroueh@us.ibm.com

Abstract

Gradient flows are a powerful tool for optimizing functionals in general metric spaces, including the space of probabilities endowed with the Wasserstein metric. A typical approach to solving this optimization problem relies on its connection to the dynamic formulation of optimal transport and the celebrated Jordan-Kinderlehrer-Otto (JKO) scheme. However, this formulation involves optimization over convex functions, which is challenging, especially in high dimensions. In this work, we propose an approach that relies on the recently introduced input-convex neural networks (ICNN) to parameterize the space of convex functions in order to approximate the JKO scheme, as well as in designing functionals over measures that enjoy convergence guarantees. We derive a computationally efficient implementation of this JKO-ICNN framework and use various experiments to demonstrate its feasibility and validity in approximating solutions of low-dimensional partial differential equations with known solutions. We also explore the use of our JKO-ICNN approach in high dimensions with an experiment in controlled generation for molecular discovery.

1 Introduction

Numerous problems in machine learning and statistics can be formulated as finding a probability distribution that minimizes some objective function of interest. One recent example of this formulation is generative modeling, where one seeks to model a data-generating distribution ρ_{data} by finding, among a parametric family ρ_θ, the distribution that minimizes some notion of discrepancy to ρ_{data}, i.e., $\min_\theta D(\rho_\theta, \rho_{\text{data}})$. Different choices of discrepancies give rise to various training paradigms, such as generative adversarial networks [23] (Jensen-Shannon divergence), Wasserstein GAN [3] (1-Wasserstein distance) and maximum likelihood estimation (KL divergence) [29, 35, 43]. In general, such problems can be cast as finding $\rho^* = \operatorname{argmin}_\rho F(\rho)$, for a functional F on distributions.

Beyond machine learning and statistics, optimization on the space of probability distributions is prominent in applied mathematics, particularly in the study of partial differential equations (PDE). The seminal work of Jordan et al. [28], and later Otto [37], Ambrosio et al. [1], and several others, showed that many classic PDEs can be understood as minimizing certain functionals defined on distributions. Central to these works is the notion of *gradient flows* on the probability space endowed with the Wasserstein metric. Jordan, Kinderlehrer, and Otto [28] set the foundations of a theory establishing connections between optimal transport, gradient flows, and differential equations. In addition, they proposed a general iterative method, popularly referred to as the JKO scheme, to solve PDEs of the Fokker-Planck type. This method was later extended to more general PDEs and in turn to more general functionals over probability space [1]. The JKO scheme can be seen as a generalization of the implicit Euler method on the probability space endowed with the Wasserstein metric. This approach has various appealing theoretical convergence properties owing to a notion of convexity of probability functionals, known as geodesic convexity (see the excellent monograph of Santambrogio [46] for more details).

Preprint. Under review.

Several computational approaches to JKO have been proposed, among them an elegant method introduced in [7] that reformulates the JKO variational problem on probability measures as an optimization problem on the space of convex functions. This reformulation is made possible thanks to Brenier's Theorem [9]. However, the appeal of this computational scheme comes at a price: computing updates involves solving an optimization over convex functions at each step, which is challenging in general. The practical implementations in [7] make use of space discretization to solve this optimization problem, which limits their applicability beyond two dimensions.

In this work, we propose a computational approach to the JKO scheme that is scalable in high-dimensions.At the core of our approach are Input-Convex Neural Networks (ICNN) [2], a recently proposed class of deep models that are convex with respect to their inputs. We use ICNNs to find parametric solutions to Benamou et al.'s [7] reformulation of the JKO problem as optimization on the space of convex functions. This leads to an approximation of the JKO scheme that we call JKO-ICNN. In practice, we implement JKO-ICNN with finite samples from distributions and optimize the parameters of ICNNs with adaptive gradient descent using automatic differentiation.

To evaluate the soundness of our approach, we first conduct experiments on well-known PDEs in low dimensions that have exact analytic solutions, allowing us to quantify the approximation quality of the gradient flows evolved with our method. We then use our approach in a high-dimensional setting, where we optimize a dataset of molecules to satisfy certain properties, such as drug-likeness (QED). The results show that our JKO-ICNN approach is successful at approximating solutions of PDEs and has the unique advantage of scalability in terms of optimizing generic probability functionals on the probability space in high dimensions.

2 Background

Notation Let $\mathcal{X}$ be a Polish space equipped with metric d, and $\mathcal{P}(\mathcal{X})$ the set of non-negative Borel measures with finite second-order moment on that space. The space $\mathcal{P}(\mathcal{X})$ contains both continuous and discrete measures, the latter represented as an empirical distribution: $\sum_{i=1}^{N} p_i \delta_{x_i}$, where δ_x is a Dirac at position $x \in \mathcal{X}$. For a measure ρ and measurable map $T : \mathcal{X} \to \mathcal{X}$, we use $T_\sharp \rho$ to denote the push-forward measure, and $\mathbf{J}_T$ the Jacobian of T. For a function $u : \mathcal{X} \to \mathbb{R}$, ∇u is the gradient and $\mathbf{H}_u(x)$ is the Hessian. $\nabla \cdot$ denotes the divergence operator. For a matrix A, $|A|$ denotes its determinant. When clear from the context, we use ρ interchangeably to denote a measure and its density.

Gradient flows in Wasserstein space Consider first a functional $F : \mathcal{X} \to \mathbb{R}$ and a point $x_0 \in \mathcal{X}$. A *gradient flow* is an absolutely continuous curve $x(t)$ that evolves from x_0 in the direction of steepest descent of F. When $\mathcal{X}$ is Hilbertian and F is sufficiently smooth, its gradient flow can be succinctly expressed as the solution of a differential equation $x'(t) = -\nabla F(x(t))$, with initial condition $x(0) = x_0$.

Gradient flows can be defined in probability space too, as long as a suitable notion of distance between probability distributions is chosen. Formally, let us consider $\mathcal{P}(\mathcal{X})$ equipped with the p-Wasserstein distance, which for measures $\alpha, \beta \in \mathcal{P}(\mathcal{X})$ is defined as:

$$\mathrm{W}_p(\alpha, \beta) \triangleq \min_{\pi \in \Pi(\alpha,\beta)} \int \|x - y\|_2^2 \, \mathrm{d}\pi(x, y). \tag{1}$$

Here $\Pi(\alpha, \beta)$ is the set of couplings (*transportation plans*) between α and β, formally: $\Pi(\alpha, \beta) \triangleq \{\pi \in \mathcal{P}(\mathcal{X} \times \mathcal{X}) \mid P_{1\sharp}\pi = \alpha, P_{2\sharp}\pi = \beta\}$. Endowed with this metric, the *Wasserstein space* $\mathbb{W}_p(\mathcal{X}) = (\mathcal{P}(\mathcal{X}), \mathrm{W}_p)$ is a complete and separable metric space. In this case, given a functional in probability space $F : \mathcal{P}(\mathcal{X}) \to \mathbb{R}$, its gradient flow in $\mathbb{W}_p(\mathcal{X})$ is a curve $\rho(t) : \mathbb{R}_+ \to \mathcal{P}(\mathcal{X})$ that satisfies $\partial_t \rho(t) = -\nabla_{\mathbb{W}_2} F(\rho(t))$. Here $\nabla_{\mathbb{W}_2}$ is a natural notion of gradient in $\mathbb{W}_p(\mathcal{X})$ given by: $\nabla_{\mathbb{W}_2} F(\rho) = -\nabla \cdot \left(\rho \nabla \frac{\delta F}{\delta \rho}\right)$ where $\frac{\delta F}{\delta \rho}$ is the first variation of the functional F. Therefore, the gradient flow of F solves the following PDE:

$$\partial_t \rho_t - \nabla \cdot \left(\rho_t \nabla \left(\frac{\delta F}{\delta \rho}(\rho_t)\right)\right) = 0 \tag{2}$$

also known as a continuity equation.

3 Gradient flows via JKO scheme and input convex neural networks

In this section, we introduce the JKO scheme for solving gradient flows, show how it can be cast as an optimization over convex functions, and propose a method to solve the resulting optimization problem via parametrization with ICNNs.

3.1 JKO scheme on probability measures

Throughout this work we consider problems of the form $\min_{\rho \in \mathcal{P}(\mathcal{X})} F(\rho)$, where $F : \mathcal{P}(\mathcal{X}) \to \mathbb{R}$ is a functional over probability measures that encodes some objective of interest. Following the gradient flow literature (e.g., [46, 47]), we will focus on three fairly general families of functionals:

$$\mathcal{F}(\rho) = \int f(\rho(x))\,\mathrm{d}x, \quad \mathcal{V}(\rho) = \int V(x)\,\mathrm{d}\rho, \quad \mathcal{W}(\rho) = \frac{1}{2}\iint W(x-x')\,\mathrm{d}\rho(x)\,\mathrm{d}\rho(x'), \tag{3}$$

where $f : \mathbb{R} \to \mathbb{R}$ is convex and superlinear and $V, W : \mathcal{X} \to \mathbb{R}$ are convex and sufficiently smooth. These functionals are appealing for various reasons. First, their gradient flows enjoy desirable convergence properties, as we discuss below. Second, they have a physical interpretation as internal, potential, and interaction energies, respectively. Finally, their corresponding continuity equation (2) turn out to recover various classic PDEs (see Table 1 for equivalences). Thus, in this work, we focus on objectives that can be written as linear combinations of these three types of functionals.

Table 1: Equivalence between gradient flows and PDEs. In each case, the gradient flow of the functional $F(\rho)$ in Wasserstein space in the rightmost column satisfies the PDE in the middle column.

Class	PDE $\partial_t \rho =$	Flow Functional $F(\rho) =$
Heat Equation	$\Delta\rho$	$\int \rho(x)\log\rho(x)\,\mathrm{d}x$
Advection	$\nabla\cdot(\rho\nabla V)$	$\int V(x)\,\mathrm{d}\rho(x)$
Fokker-Planck	$\Delta\rho + \nabla\cdot(\rho\nabla V)$	$\int \rho(x)\log\rho(x)\,\mathrm{d}x + \int V(x)\,\mathrm{d}\rho(x)$
Porous Media	$\Delta(\rho^m) + \nabla\cdot(\rho\nabla V)$	$\frac{1}{m-1}\int \rho(x)^m\,\mathrm{d}x + \int V(x)\,\mathrm{d}\rho(x)$
Adv. + Diff. + Interaction	$\nabla\cdot\left[\rho(\nabla f'(\rho) + \nabla V + (\nabla W) * \rho)\right]$	$\int V(x)\,\mathrm{d}\rho(x) + \int f(\rho(x))\,\mathrm{d}x + \frac{1}{2}\iint W(x-x')\,\mathrm{d}\rho(x)\,\mathrm{d}\rho(x')$

For a functional F of this form, it can be shown that the corresponding gradient flow defined in Section 2 converges exponentially fast to a unique minimizer [46]. This suggests solving the optimization problem $\min_\rho F(\rho)$ by following the gradient flow, starting from some initial configuration ρ_0. A convenient method to study this PDE is through the time discretization provided by the Jordan–Kinderlehrer–Otto (JKO) iterated movement minimization scheme [28]:

$$\rho^\tau_{t+1} \in \operatorname*{argmin}_{\rho \in \mathbb{W}_2(\mathcal{X})} F(\rho) + \frac{1}{2\tau}\mathrm{W}_2^2(\rho, \rho^\tau_t), \tag{4}$$

where $\tau > 0$ is a time step parameter. This scheme will form the backbone of our approach.

3.2 From measures to convex functions

The general JKO scheme (4) discretizes the gradient flow (and therefore, the corresponding PDE) in time, but it is still formulated on —potentially infinite-dimensional, and therefore intractable— probability space $\mathcal{P}(\mathcal{X})$. Obtaining an implementable algorithm requires recasting this optimization problem in terms of a space that is easier to handle than that of probability measures. As a first step, we do so using convex functions.

One of the cornerstones of optimal transport theory states that for absolutely continuous measures and suitable cost functions, the solution of the Kantorovich problem concentrates around a deterministic map T (the Monge map). Furthermore, for the quadratic cost, Brenier's theorem [9] states that this map is given by the gradient of a convex function u, i.e., $T(x) = \nabla u(x)$. Hence given a measure α, the mapping $u \in \mathrm{cvx}(\mathcal{X}) \mapsto (\nabla u)_\sharp \alpha \in \mathcal{P}(\mathcal{X})$ can be seen as a parametrization, which depends on α, of the space of probabilities [34]. We furthermore have for any $u \in \mathrm{cvx}(\mathcal{X})$:

$$\mathrm{W}_2^2(\alpha, (\nabla u)_\sharp \alpha)) = \int_{\mathcal{X}} \|\nabla u(x) - x\|_2^2\,\mathrm{d}\alpha. \tag{5}$$

Using this expression and the parametrization $\rho = (\nabla u)_\sharp \rho^\tau_t$ with $u \in \mathrm{cvx}(\mathcal{X})$ in Problem 4 we obtain a reformulation of Wasserstein gradient flows as optimization over convex functions [7]:

$$u^\tau_{t+1} \in \operatorname*{argmin}_{u \in \mathrm{cvx}(\mathcal{X})} F((\nabla u)_\sharp \rho^\tau_t) + \frac{1}{2\tau}\int_{\mathcal{X}} \|\nabla u(x) - x\|_2^2\,\mathrm{d}\rho^\tau_t, \tag{6}$$

which implicitly defines a sequence of measures via $\rho_{t+1}^\tau = (\nabla u_{t+1}^\tau)_\#(\rho_t^\tau)$. For potential and interaction functionals, Lemma 3.1 shows that the first term in this scheme can be written in a form amenable to optimization on u.

Lemma 3.1 (Potential and Interaction Energies)**.** *Let ρ_t be the measure at time t of the JKO iterations. For the pushforward measure $\rho = (\nabla u)_\sharp \rho_t$, the functionals $\mathcal{V}$ and $\mathcal{W}$ can be written as:*

$$\mathcal{V}(\rho) = \int (V \circ \nabla u)(x)\, \mathrm{d}\rho_t(x) \quad \textit{and} \quad \mathcal{W}(\rho) = \frac{1}{2} \iint W(\nabla u(x) - \nabla u(y))\, \mathrm{d}\rho_t(y)\, \mathrm{d}\rho_t(x). \tag{7}$$

Crucially, ρ_t appears here only as the integrating measure. We will exploit this property for finite-sample computation in the next section. In the case of internal energies $\mathcal{F}$, however, the integrand itself depends on ρ_t, which poses difficulties for computation. To address this, we start in Lemma 3.2 by tackling the change of density when using strictly convex potential pushforward maps:

Lemma 3.2 (Change of Variable)**.** *Given a strictly convex potential $u \in \mathrm{cvx}(\mathcal{X})$, ∇u is invertible, and $(\nabla u)^{-1} = \nabla u^*$, where u^* is the convex conjugate of u, $u^*(y) = \sup_{x \in dom(u)} \langle x, y\rangle - u(x)$. Given a measure α with density ρ_α, the density ρ_β of the measure $\beta = (\nabla u)_\sharp \alpha$ is given by:*

$$\rho_\beta(y) = \frac{\rho_\alpha}{|\mathbf{H}_u|} \circ (\nabla u)^{-1}(y) = \frac{\rho_\alpha}{|\mathbf{H}_u|} \circ \nabla u^*(y). \tag{8}$$

In other words $\log\big(\rho_\beta(y)\big) = \log\big(\rho_\alpha(\nabla u^*(y))\big) - \log(|\mathbf{H}_u(\nabla u^*(y))|)$. Iterating Lemma 3.2 across time in the JKO steps we obtain:

Corollary 3.3 (Iterated Change of Variables in JKO)**.** *Assume ρ_0 has a density. Let $T_{1:t}^\tau = \nabla u_t^\tau \cdots \circ \nabla u_1^\tau$, where u_t^τ are optimal convex potentials in the JKO sequence that we assume are strictly convex. We use the convention $T_{1:0}^\tau(x) = x$. We have $(T_{1:t}^\tau)^{-1} = \nabla (u_1^\tau)^* \circ \cdots \circ \nabla (u_t^\tau)^*$ where $(u_t^\tau)^*$ is the convex conjugate of u_t^τ. At time t of the JKO iterations we have: $\rho_t(x) = T_{1:t}^\tau \rho_0(x)$, and therefore:*

$$\log(\rho_t^\tau) = \left(\log(\rho_0) - \sum_{s=1}^{t} \log(|\mathbf{H}_{u_s^\tau}(T_{1:s-1}^\tau)|) \right) \circ (T_{1:t}^\tau)^{-1}. \tag{9}$$

From Corollary 3.3, we see that the iterates in the JKO scheme imply a change of densities that shares similarities with normalizing flows [26, 43], where the depth of the flow network in [43] corresponds to the time in JKO. Whereas the normalizing flows in [43] draw connections to the Fokker-Planck equation in the generative modeling context, JKO is more general and allows for rigorous optimization of generic functionals on the probability space.

Armed with this expression of ρ_t^τ, we can now write $\mathcal{F}$ in terms of the convex potential u:

Lemma 3.4 (Internal Energy)**.** *Let ρ_t be the measure at the time t of the JKO iterations. Using notations of Corollary 3.3, for the pushforward measure $\rho = (\nabla u)_\sharp \rho_t = (\nabla u \circ T_{1:t})_\sharp \rho_0$ we have:*

$$\mathcal{F}(\rho) = \int f\left(\frac{\rho_0(x)}{|\mathbf{H}_u(T_{1:t}(x))||\mathbf{J}_{T_{1:t}}(x)|} \right) |\mathbf{H}_u(T_{1:t}(x))||\mathbf{J}_{T_{1:t}}(x)|\, \mathrm{d}x \tag{10}$$

Letting $\xi(x) = \frac{\rho_0(x)}{|\mathbf{H}_u(T_{1:t}(x))||\mathbf{J}_{T_{1:t}}(x)|}$ and assuming $\rho_0 > 0$, we have: $\mathcal{F}(\rho) = \mathbb{E}_{x\sim\rho_0} \frac{f\circ\xi}{\xi}(x)$.

3.3 From convex functions to finite parameters

Solving problem (6) requires: (i) a tractable parametrization of $\mathrm{cvx}(\mathcal{X})$, the space of convex functions, (ii) a method to evaluate and compute gradients of the Wasserstein distance term, and (iii) a method to evaluate and compute gradients of the functionals as expressed in Lemmas 3.1 and 3.4.

For (i), we rely on the recently proposed Input Convex Neural Networks [2]. See Appendix B.1 for a background on ICNN. Given $\rho_0 = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ we solve for $t = 1 \dots T$:

$$\theta_{t+1}^\tau \in \underset{\theta:\, u_\theta \in \mathrm{ICNN}(\mathcal{X})}{\operatorname{argmin}} L(\theta) \triangleq F((\nabla_x u_\theta(x))_\sharp \rho_t^\tau) + \frac{1}{2\tau} \int_{\mathcal{X}} \|\nabla_x u_\theta(x) - x\|_2^2\, \mathrm{d}\rho_t^\tau, \tag{11}$$

where $\mathrm{ICNN}(\mathcal{X})$ is the space of Input Convex Neural Networks, and the θ denotes parameters of the ICNN. Equation (11) defines a JKO sequence of measures via $\rho_{t+1}^\tau = (\nabla_x u_{\theta_{t+1}^\tau})_\#(\rho_t^\tau)$. We call this

iterative process JKO-ICNN, where each optimization problem can be solved with gradient descent on the parameter space of the ICNN, using backpropagation and automatic differentiation.

For (ii), we note that this term can be interpreted as an expectation, namely, $\frac{1}{2\tau}\,\mathbb{E}_{x\sim\rho_t^\tau}\,\|\nabla_x u_\theta(x) - x\|_2^2$, so we can approximate it with finite samples (*particles*) of ρ_t^τ obtained via the pushforward map of previous cloud points in JKO sequence: $\frac{1}{2\tau n}\sum_{i=1}^n \|\nabla_x u_\theta(x_i) - x_i\|_2^2, \{x_i\}_{i=1}^n \sim \rho_t^\tau$. Finally, for (iii) we first note that the $\mathcal{V}$ and $\mathcal{W}$ functionals can also be written as expectations over ρ_t^τ:

$$\mathcal{V}\big((\nabla_x u_\theta)_\sharp \rho_t^\tau\big) = \mathop{\mathbb{E}}_{x\sim\rho_t^\tau} V(\nabla_x u_\theta(x)), \mathcal{W}\big((\nabla_x u_\theta)_\sharp \rho_t^\tau\big) = \tfrac{1}{2}\mathop{\mathbb{E}}_{x,y\sim\rho_t^\tau} W(\nabla_x u_\theta(x) - \nabla_x u_\theta(y)). \quad (12)$$

Thus, as long as we can parametrize the functions V and W in a differentiable manner, we can estimate the value and gradients of these two functionals through finite samples too. In many cases V and W will be simple analytic functions, such as in the PDEs considered in Section 5. To model more complex optimization objectives with these functionals we can leverage ICNNs once more to parametrize the functions V and W as neural networks in a way that enforces their convexity. This is what we do in the molecular discovery experiments in Section 6.

Particular cases of internal energies Equation (10) simplifies for some choices of f. For example, for $f(t) = t\log t$ (which yields the heat equation) and assuming strict convexity of u we get:

$$\mathcal{F}\big((\nabla u_\theta)_\sharp \rho_t\big) = \int \frac{\rho_t(x)}{|\mathbf{H}_{u_\theta}(x)|}\log\frac{\rho_t(x)}{|\mathbf{H}_{u_\theta}(x)|}\big|\mathbf{H}_{u_\theta}(x)\big|\,\mathrm{d}x = \mathcal{F}(\rho_t) - \int \log|\mathbf{H}_{u_\theta}(x)|\rho_t(x)\,\mathrm{d}x \quad (13)$$

where we drop τ from the notation for simplicity. This expression has a notable interpretation: pushing forward measure ρ_t by ∇u_θ increases its entropy by a log-determinant barrier term on u_θ's Hessian. Note that only the second term in Equation (13) depends on u_θ, and that it can be approximated —as in the previous cases— by an empirical expectation, so $\nabla_\theta \mathcal{F}\big((\nabla_x u_\theta)_\sharp \rho_t\big) = -\nabla_\theta \mathbb{E}_{x\sim\rho_t}\log|\mathbf{H}_{u_\theta}(x)|$. This suggests using the latter as a surrogate objective for optimization.

Another notable case is $f(t) = \frac{1}{m-1}t^m, m > 1$, which yields a nonlinear diffusion term as in the porous medium equation (Table 1). In this case, the expression in Lemma 3.4 takes the form $\mathcal{F}\big(\rho\big) = \frac{1}{m-1}\mathbb{E}_{x\sim\rho_0}\xi(x)^{m-1}$, which has following gradient w.r.t. θ:

$$-\mathop{\mathbb{E}}_{x\sim\rho_0}\exp\{(m-1)\log\xi(x)\}\nabla_\theta\log|\mathbf{H}_{u_\theta}(T_{1:t}(x))| \quad (14)$$

All surrogate objectives are summarized in Table 3 in the Appendix.

Algorithm 1: JKO-ICNN: JKO variational scheme using input convex neural networks

---

Input: $F(\rho)$ to optimize ,$\tau > 0$ JKO learning rate (outer loop), $\eta > 0$ Learning rate for ICNN (inner loop), n_u number of iterations the inner loop, T number of JKO steps, *warmstart* boolean
Initialize $u_\theta \in$ ICNN, θ parameters of ICNN, $\rho_0^\tau = \frac{1}{N}\sum_{i=1}^N \delta_{x_i}$.
for $t = 0$ **to** $T-1$ **do**
  if not *warmstart*: $\theta \leftarrow$ InitializeWeights()
  for $i = 1$ **to** n_u **do**
    {JKO inner loop: Updating ICNN }
    $L(\theta) = F((\nabla_x u_\theta)_\# \rho_t^\tau) + \frac{1}{2\tau}\mathbb{E}_{\rho_t^\tau}||x - \nabla_x u_\theta(x)||^2$
    $\theta \leftarrow \mathrm{Adam}(\theta, \eta, \nabla_\theta L_\theta)$
  end for
  {JKO outer loop: Updating point cloud (measures) }
  $\rho_{t+1}^\tau = \nabla_x(u_\theta)_\# \rho_t^\tau$
end for
Output: ρ_T^τ

Implementation and practical considerations We implement the proposed JKO-ICNN scheme (Algorithm 1) in pytorch [38], relying on automatic differentiation to solve the inner optimization loop. The finite-sample approximations of $\mathcal{V}$ and $\mathcal{W}$ (Eq. (12)) can be used directly, but the computation of the surrogate objectives for internal energies $\mathcal{F}$ (i.e., Eqs. (13), (14)) require computing Hessian log-determinants, which is prohibitive in high dimensions. Thus, we use a stochastic log-trace

estimator based on the Hutchinson method [27], as used by Huang et al. [26] (see §B.4 for details). To enforce strong convexity on the ICNN, we clip its weights to positive values after each update. When needed (e.g., for evaluation), we estimate the true internal energy functional $\mathcal{F}$ by using Corollary 3.3 to compute densities. We provide further implementation details in Appendix B.5.

4 Related Work

Computational gradient flows Gradient flows have been implemented through various computational methods. Benamou et al. [6] propose an augmented Lagrangian approach for convex functionals based on the dynamical optimal transport implementation of Benamou and Brenier [5]. Another approach relying on the dynamic formulation of JKO and an Eulerian discretization of measures (i.e. via histograms) is the recent primal dual algorithm of Carrillo et al. [12]. Closer to our work is the formulation of Benamou et al. [7] that casts the problem as an optimization over convex functions. This work relies on a Lagrangian discretization of measures, via cloud points, and on a representation of convex functions and their corresponding subgradients via their evaluation at these points. This method does not scale well in high dimensions since it computes Laguerre cells in order to find the subgradients. A different approach by Peyré [40] defines entropic gradient flows using Eulerian discretization of measures and Sinkhorn-like algorithms that leverage an entropic regularization of the Wasserstein distance. Frogner and Poggio [21] propose kernel approximations to compute gradient flows. Finally, blob methods have been considered in Craig and Bertozzi [16] and Carrillo et al. [13] for the aggregation and diffusion equations. Blob methods regularize velocity fields with mollifiers (convolution with a kernel) and allow for the approximation of internal energies.

ICNN, optimal transport, and generative modeling ICNN architectures were originally proposed by Amos et al. [2] to allow for efficient inference in settings like structured prediction, data imputation, and reinforcement learning. Since their introduction, they have been exploited in various other settings that require parametrizing convex functions, including optimal transport. For example, Makkuva et al. [33] propose using them to learn an explicit optimal transport map between distributions, which under suitable assumptions can be shown to be the gradient of a convex function [9]. The ICNN parametrization has been also exploited in order to learn continuous Wasserstein barycenters by Korotin et al. [30]. Using this same characterization, Huang et al. [26] recently proposed to use ICNNs to parameterize flow-based invertible probabilistic models, an approach they call *convex potential flows*. These type of flows, popularized in the normalizing flow literature and which should not be confused with *gradient flows*, are useful for learning generative models when samples from the target (i.e., optimal) distributions are available and when the goal is to to learn a parametric generative model. Our use of ICNNs differs from these prior works and other approaches to generative modeling. First, we consider the setting where no samples from the target distribution are given; instead, only a functional characterization of this distribution is provided, i.e., as a minimizer of an optimization problem over distributions. Additionally, in terms of usage, we leverage ICNNs not for solving a single optimal transport problem, but rather for a sequence of JKO step optimization problems that involve various terms, one of which is a Wasserstein distance.

5 Evolving PDEs with known solutions

We first evaluate our method on gradient flows whose corresponding PDEs have known solutions. We focus on three examples from Carrillo et al. [12] that combine the three types of functionals considered here (§3): porous medium, non-linear Fokker-Planck, and aggregation equations. Throughout this section, we use our JKO-ICNN with $\tau = \eta = 10^{-3}$ in the notation of Algorithm 1.

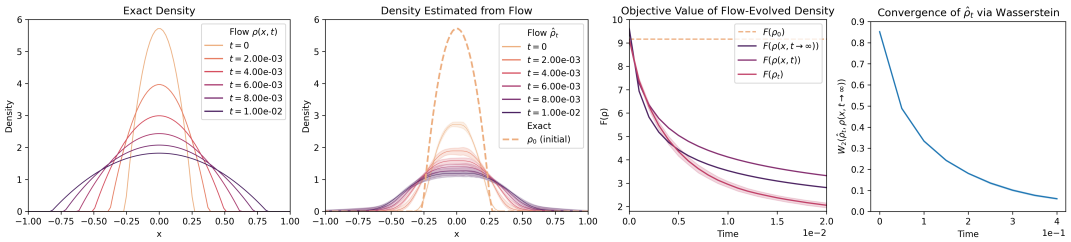
5.1 Porous medium equation on Barenblatt-Pattle profiles

The porous medium equation is a classic non-linear diffusion PDE. We consider in particular a diffusion-only system: $\partial_t \rho = \Delta \rho^m, m > 1$, corresponding to a gradient flow of the internal energy functional $\mathcal{F}(\rho) = \frac{1}{m-1} \int \rho^m(x)\,\mathrm{d}x$, which we implement using our JKO-ICNN with objective (14). A known family of exact solutions of this PDE is given by Barenblatt-Pattle profiles [4, 39, 52]:

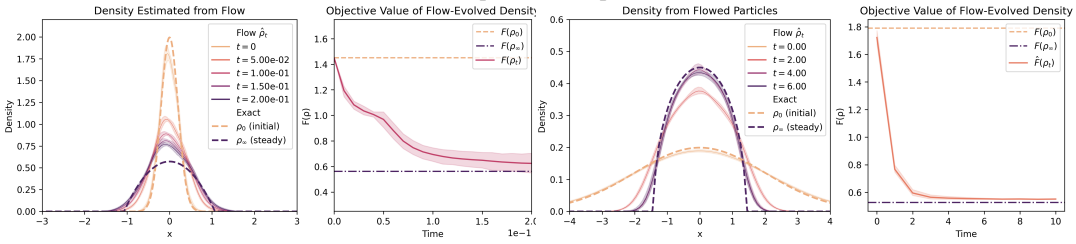
$$\rho(x,t) = t^{-\alpha}\left(C - k\|x\|^2 t^{-2\beta}\right)_+^{\frac{1}{m-1}}, \qquad x \in \mathbb{R}^d,\ t > 0,$$

where $C > 0$ is a constant and $\alpha = d/(d(m-1)+2)$, $\beta = \alpha/d$, and $k = \alpha(m-1)/(2md)$.

This exact solution provides a trajectory of densities to compare our JKO-ICNN approach against. Specifically, starting from particles sampled from $\rho(x,0)$, we can compare the trajectory $\hat{\rho}_t(x)$



(a) Porous medium equation with pure diffusion §5.1.



(b) Fokker-Planck with nonlinear diffusion §5.2. **(c)** Aggregation equation §5.3.

Figure 1: Flows on PDEs with known solution. We use KDE on the flowed particles for the density plots, and the iterated push-forward density method (Corollary 3.3) to evaluate $\mathcal{F}$ in (a) and (b).

estimated with our method to the exact density $\rho(x,t)$. Although this system has no steady-state solution, its asymptotic behavior can be expressed analytically too. For the case $d=1, m=2, C = (3/16)^{1/3}$, Figure 1a shows that our method is able to reproduce the dynamics of the exact solution (here the flow density is estimated from particles via KDE and aggregated over 10 repetitions with random initialization), and that the objective value $\mathcal{F}(\hat{\rho})$ has the correct asymptotic behavior.

5.2 Nonlinear Fokker-Planck equation

Next, we consider a Fokker-Planck equation with a non-linear diffusion term as before:

$$\partial_t \rho = \nabla \cdot (\rho \nabla V) + \Delta \rho^m, \quad V : \mathbb{R}^d \to \mathbb{R}, \quad m > 1. \tag{15}$$

This PDE corresponds to a gradient flow of the objective $F(\rho) = \frac{1}{m-1} \int \rho^m(x)\,\mathrm{d}x + \int V(x)\,\mathrm{d}\rho(x)$. For certain choices of V, the solutions of this equation approach a unique steady state [10]:

$$\rho_\infty(x) = \big(C - \tfrac{m-1}{m} V(x)\big)_+^{\frac{1}{m-1}}, \tag{16}$$

where the constant C depends on the initial mass of the data. For $d=1$, $m=2$, and $V(x)=x^2$, we solve this PDE using JKO-ICNN with objectives (12) and (14), using initial data drawn from a Normal distribution with parameters $(\mu, \sigma^2) = (0, 0.2)$. Unlike the previous example, in this case we do not have a full solution $\rho(x,t)$ to compare against, but we can instead evaluate convergence of the flow to $\rho_\infty(x)$. Figure 1b shows that the density $\hat{\rho}_t(x)$ derived from the JKO-ICNN flow converges to the steady state $\rho_\infty(x)$, and so does the value of the objective, i.e., $F(\hat{\rho}_t) \to F(\rho_\infty)$.

5.3 Aggregation equation

Next, we consider an aggregation equation: $\partial_t \rho = \nabla \cdot (\rho \nabla W * \rho), W : \mathbb{R}^d \to \mathbb{R}$, which corresponds to a gradient flow on an interaction functional $\mathcal{W}(\rho) = \frac{1}{2} \iint W(x-x')\,\mathrm{d}\rho(x)\,\mathrm{d}\rho(x')$. We consider the same setting as Carrillo et al. [12]: $d=1$, $\rho_0 \sim \mathcal{N}(0,1)$, and the kernel $W(x) = \frac{1}{2}|x|^2 - \log(|x|)$, which enforces repulsion at short length scales and attraction at longer scales. This choice of W has the advantage of yielding a unique steady-state equilibrium [11], given by $\rho_\infty(x) = \frac{1}{\pi}\sqrt{(2-x^2)_+}$. Our JKO-ICNN encodes $\mathcal{W}$ using the objective (12). As in the previous section, we investigate the convergence of this flow to this steady state distribution. Figure 1c shows that in this case too we observe convergence of densities $\hat{\rho}_t(x) \to \rho_\infty(x)$ and objective values $F(\hat{\rho}_t) \to F(\rho_\infty)$.

6 Molecular discovery with JKO-ICNN

To demonstrate the flexibility and efficacy of our approach, we apply it in an important high dimensional setting: controlled generation in molecular discovery. For the reader less familiar with molecular generation, this application is analogous to conditional image generation and editing. In our experiments, our goal is to increase the *drug-likeness* of a given distribution of molecules while

staying close to the original distribution, an important task in drug discovery and drug re-purposing. Formally, given an initial distributions of molecules ρ_0 and a convex potential energy function $V(\cdot)$ that models the property of interest, the functional we wish to minimize is:

$$\min_{\rho \in \mathcal{P}(\mathcal{X})} F(\rho) := \lambda_1 \mathbb{E}_\rho V(x) + \lambda_2 \mathrm{D}(\rho, \rho_0),$$

where D is a divergence. In particular, we use either the 2-Wasserstein distance with entropic regularization [17] or the Maximum Mean Discrepancy [24] with a Gaussian kernel (MMD). Finally, we use our JKO-ICNN scheme (Algorithm 1) to optimize this functional on the space of probability measures, given an initial distribution $\rho_0(x) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i}(x)$ where x_i is a molecular embedding.

In what follows, we show how we model each component of this functional via : i) training a molecular embedding using a Variational Auto-encoder (VAE), ii) training a surrogate potential V to model the drug-likeness, iii) using automatic differentiation via the divergence D.

Embedding of molecules using VAEs We start by training a VAE, which is a generative model that aims to reconstruct molecule representations but is trained with a regularization term that ensures smoothness of the encoder's latent space [25, 29]. The VAE is trained on a string representation of molecules [15, 41] known as SMILES [51]. We train the VAE on a molecular dataset known as MOSES [41], which is a subset of the ZINC database [48], released under the MIT license. This dataset contains about 1.6M training and 176k test molecules. Given a molecule, we embed it using the encoder of the VAE and represent it with a vector $x_i \in \mathbb{R}^{128}$. See Appendix E, where we also present results for the experiment run on the popular QM9 dataset [42, 45].

Training a convex surrogate for the desired property (high QED) The quantitative estimate of drug-likeness (QED) [8] can be computed with the RDKit library [31, 32] but is not differentiable nor convex. Hence, we propose to learn a convex surrogate using Residual ICNNs [2, 26]. This ensures that it can be used as convex potential functional $\mathcal{V}$, as described in Section 3. To do so we process the MOSES dataset via RDKit and obtain a labeled set with all QED values. We set a QED threshold of 0.85 and give a lower value label for all QED values above that threshold and a higher value label for all QED values below it so that minimizing the potential functional with this convex surrogate will lead to higher QED values. Given VAE embeddings of the molecules, we train a ICNN classifier on this dataset. See Appendix D for experimental details.

Automatic differentiation via D When D is the entropy-regularized Wasserstein distance, we use the Sinkhorn algorithm [17] to compute it (and henceforth denote it as Sinkhorn). We backpropagate through this objective as proposed by Genevay et al. [22], using the `geomloss` toolbox for efficiency [19]. We also use `geomloss` for evaluation and backpropagation when D is chosen to be MMD.

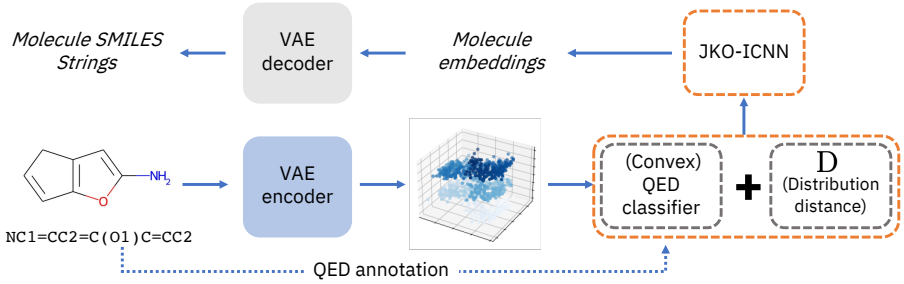


Figure 2: Molecular discovery Setup. We apply JKO-ICNN to the controlled molecular generation problem. Drug likeness and closeness to ρ_0 are maximized in the encoding space of a VAE.

Optimization with JKO With the molecule embeddings coming from the VAE serving as the point cloud to be transported and the potential functional defined by the convex QED classifier, we run Algorithm 1 to move an initial point cloud of molecule embeddings ρ_0 with low drug-likeness (QED < 0.7) to a region of the latent space that decodes to molecules with distribution ρ_τ^T with higher drug-likeness. The divergence D between the initial point cloud and subsequent point clouds of embeddings allows us to control for other generative priorities, such as staying *close* to the original set of molecules. Following the notation of Algorithm 1, we used the following hyperparameters for the JKO-ICNN in this setting. N, number of original embeddings, was 1,000. The JKO rate τ was set to $1\mathrm{e}^{-4}$ and the outer loop steps T was set to 100. For the inner loop, the number of iterations n_u

was set to 500, and the inner loop learning rate η was set to $1e^{-3}$. For the JKO ICNN, we used a fully-connected ICNN with two hidden layers, each of dimension 100. Finally, we ran the JKO-ICNN flow without warm starts between steps. The full pipeline for this experiment setting is displayed in Figure 2. All computation for this pipeline, was done in a compute environment with 1 CPU and 1 V100 GPU submitted as resource-restricted jobs to a cluster.

Table 2: Molecular discovery results. Measures of validity, uniqueness, and median QED are reported in each row for the corresponding point cloud of embeddings. The first row shows these metrics for the initial dataset, and the subsequent rows correspond to the datasets obtained with JKO-ICNN for different D (Sinkhorn or MMD) and weight parameter λ_2. Each measurement value cell contains mean values $\pm$ one standard deviation for 5 repeated runs with different random initialization seeds. Using Sinkhorn with weight 10,000 yields improved QED (higher is better) without sacrificing validity and uniqueness of the decoded SMILES strings.

Measure	D	λ_2	Validity	Uniqueness	QED Median
ρ_0	N/A	N/A	100.000 ± 0.000	99.980 ± 0.045	0.630 ± 0.001
ρ_T^τ	Sinkhorn	1,000	92.460 ± 2.096	69.919 ± 4.906	0.746 ± 0.016
ρ_T^τ	Sinkhorn	10,000	93.020 ± 1.001	99.245 ± 0.439	0.769 ± 0.002
ρ_T^τ	MMD	1,000	94.560 ± 1.372	51.668 ± 2.205	0.780 ± 0.009
ρ_T^τ	MMD	10,000	92.020 ± 3.535	53.774 ± 3.013	0.776 ± 0.014

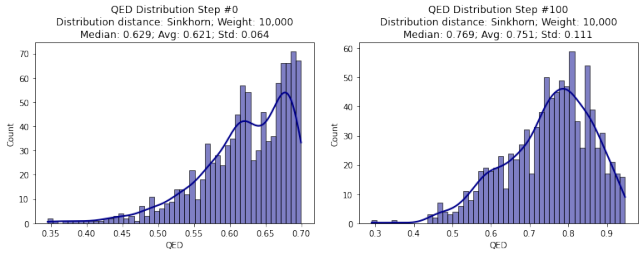


Figure 3: Histograms of QED values for the decoded SMILES strings corresponding to $t = 0$ and $t = T = 100$ for the JKO-ICNN experiment (D $=$ Sinkhorn, λ_2=10,000). We observe a shift to the right in the QED distribution, which corresponds to increased drug-likeness of the decoded molecule.

Evaluation We fixed $\lambda_1 = 1$ for all experiments, and used either Sinkhorn or MMD for D. The weight λ_2 on D was set to either 1,000 or 10,000. We start JKO with an initial cloud point ρ_0 of embedding that have QED < 0.7 randomly sampled from the MOSES test set. In the Table 2, we report several measurements. First is validity, which is the proportion of the decoded embeddings that have valid SMILES strings according to RDKit. Of the valid strings, we calculate the percent that are unique. Finally, we use RDKit to get the QED annotation of the decoded embeddings and report median values for the point cloud. We re-ran experiments five times with different random seed initializations and report means and standard deviations. In the first row of Table 2, we report the initial values for the point cloud at time $t = 0$. In the last four rows of the table, we report the measurements for different hyperparameter configurations at the end of the JKO scheme for $t = T = 100$. We see that the JKO-ICNN is able to optimize the functional objective and leads to molecules that satisfy low energy potential, i.e. with improved drug-likeness. The results are stable across different runs as can be seen from the reported standard deviations from the different random initializations. We notice that Sinkhorn divergence with λ_2 =10,000 prevents mode collapse and preserves uniqueness of the transported embedding via JKO-ICNN. While MMD yields higher drug-likeness, it leads to a deterioration in uniqueness. Using Sinkhorn allows for a matching between the transformed point cloud and the original one, which preserves better uniqueness than MMD, which merely matches mean embeddings of the distributions. Finally, in Figure 3 we display the histograms of the QED values for the decoded SMILES strings corresponding to the point clouds at time step $t = 0$ and $t = T = 100$ for D $=$ Sinkhorn, λ_2=10,000.

7 Discussion

In this paper we proposed JKO-ICNN, a scalable method for computing Wasserstein gradient flows. Key to our approach is the parameterization of the space of convex functions with Input Convex

Neural Networks. We showed that JKO-ICNN succeeds at optimizing functionals on the space of probability distributions, both in low-dimensional settings involving known PDES, as well as in large-scale and high-dimensional experiments on molecular discovery via controlled generation. Studying the convergence of solutions of JKO-ICNN is an interesting open question that we leave for future work. The potential risks or benefits of a method of this kind depend on the area of application. For the controllable molecule discovery problem tackled here, there are clear potential societal benefits, ranging from drug discovery [49] to vaccine development [36]. To mitigate potential risks, any biochemical discoveries made with an automated system such as the one proposed in this work should be independently verified in the laboratory, *in vitro* and *in vivo*, before being deployed.

References

[1] L. Ambrosio et al. *Gradient flows in metric spaces and in the Wasserstein space of probability measures*. Lectures in Mathematics. ETH Zürich. Birkhäuser Basel, 2005.

[2] B. Amos et al. "Input Convex Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 146–155.

[3] M. Arjovsky et al. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 214–223.

[4] G. I. Barenblatt. "On some unsteady motions of a liquid and gas in a porus medium". In: *Prikl. Mat. Mekh.* 16 (1952), pp. 67–78.

[5] J.-D. Benamou and Y. Brenier. "A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem". In: *Numerische Mathematik* (2000).

[6] J.-D. Benamou et al. "An augmented Lagrangian approach to Wasserstein gradient flows and applications". In: *ESAIM: ProcS* 54 (2016), pp. 1–17.

[7] J.-D. Benamou et al. *Discretization of functionals involving the Monge-Ampère operator*. 2014. arXiv: 1408.4536 [math.NA].

[8] G. R. Bickerton et al. "Quantifying the chemical beauty of drugs". In: *Nature chemistry* 4.2 (2012), pp. 90–98.

[9] Y. Brenier. "Polar factorization and monotone rearrangement of vector-valued functions". In: *Communications on Pure and Applied Mathematics* 44.4 (1991), pp. 375–417.

[10] J. A. Carrillo and G. Toscani. "Asymptotic L^1-decay of Solutions of the Porous Medium Equation to Self-similarity". In: *Indiana Univ. Math. J.* 49.1 (2000), pp. 113–142.

[11] J. A. Carrillo et al. "A mass-transportation approach to a one dimensional fluid mechanics model with nonlocal velocity". In: *Adv. Math.* 231.1 (Sept. 2012), pp. 306–327.

[12] J. A. Carrillo et al. "Primal Dual Methods for Wasserstein Gradient Flows". In: *Found. Comut. Math.* (Mar. 2021).

[13] J. A. Carrillo et al. *A blob method for diffusion*. 2019. arXiv: 1709.09195 [math.AP].

[14] R. T. Q. Chen et al. "Residual Flows for Invertible Generative Modeling". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[15] V. Chenthamarakshan et al. "Cogmol: Target-specific and selective drug design for covid-19 using deep generative models". In: *arXiv preprint arXiv:2004.01215* (2020).

[16] K. Craig and A. L. Bertozzi. *A Blob Method for the Aggregation Equation*. 2014. arXiv: 1405.6424 [math.NA].

[17] M. Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 2292–2300.

[18] W. Falcon et al. "PyTorch Lightning". In: *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning* 3 (2019).

[19] J. Feydy et al. "Interpolating between Optimal Transport and MMD using Sinkhorn Divergences". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019, pp. 2681–2690.

[20] R. Flamary et al. “POT: Python Optimal Transport”. In: *Journal of Machine Learning Research* 22.78 (2021), pp. 1–8.

[21] C. Frogner and T. Poggio. “Approximate Inference with Wasserstein Gradient Flows”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by S. Chiappa and R. Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, Aug. 2020, pp. 2581–2590.

[22] A. Genevay et al. “Learning Generative Models with Sinkhorn Divergences”. In: *International Conference on Artificial Intelligence and Statistics*. Vol. 84. PMLR, 2018, pp. 1608–1617.

[23] I. Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014.

[24] A. Gretton et al. “A Kernel Two-sample Test”. In: *JMLR* (2012).

[25] I. Higgins et al. “beta-vae: Learning basic visual concepts with a constrained variational framework”. In: (2016).

[26] C.-W. Huang et al. “Convex Potential Flows: Universal Probability Distributions with Optimal Transport and Convex Optimization”. In: *International Conference on Learning Representations*. 2021.

[27] M. F. Hutchinson. “A Stochastic Estimator of the Trace of the Influence Matrix for Laplacian Smoothing Splines”. In: *Communications in Statistics - Simulation and Computation* 18.3 (Jan. 1989), pp. 1059–1076.

[28] R. Jordan et al. “The Variational Formulation of the Fokker–Planck Equation”. In: *SIAM J. Math. Anal.* 29.1 (Jan. 1998), pp. 1–17.

[29] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[30] A. Korotin et al. “Continuous Wasserstein-2 Barycenter Estimation without Minimax Optimization”. In: *International Conference on Learning Representations*. 2021.

[31] G. Landrum. *RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling*. 2013.

[32] G. Landrum. *RDKit: Open-source cheminformatics*.

[33] A. Makkuva et al. “Optimal transport mapping via input convex neural networks”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. I. A. Singh. Vol. 119. PMLR, 2020, pp. 6672–6681.

[34] R. J. McCann. “A Convexity Principle for Interacting Gases”. In: *Advances in Mathematics* 128.1 (1997), pp. 153–179.

[35] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series. MIT Press, 2012.

[36] E. Ong et al. “COVID-19 coronavirus vaccine design using reverse vaccinology and machine learning”. en. In: *Front. Immunol.* (July 2020).

[37] F. Otto. “The geometry of dissipative evolution equations: the porous medium equation”. In: *Comm. Partial Differential Equations* 26.1-2 (Jan. 2001), pp. 101–174.

[38] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[39] R. E. Pattle. “Diffusion from an instantaneous point source with a concentration-dependent coefficient”. en. In: *Quart. J. Mech. Appl. Math.* 12.4 (Jan. 1959), pp. 407–409.

[40] G. Peyré. “Entropic Approximation of Wasserstein Gradient Flows”. In: *SIAM J. Imaging Sci.* 8.4 (Jan. 2015), pp. 2323–2351.

[41] D. Polykovskiy et al. “Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models”. In: *Frontiers in Pharmacology* (2020).

[42] R. Ramakrishnan et al. “Quantum chemistry structures and properties of 134 kilo molecules”. In: *Scientific data* 1.1 (2014), pp. 1–7.

[43] D. Rezende and S. Mohamed. “Variational Inference with Normalizing Flows”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1530–1538.

[44] R. T. Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton, N. J.: Princeton University Press, 1970.

[45] L. Ruddigkeit et al. "Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17". In: *Journal of chemical information and modeling* 52.11 (2012), pp. 2864–2875.

[46] F. Santambrogio. "{Euclidean, metric, and Wasserstein} gradient flows: an overview". In: *Bull. Math. Sci.* 7.1 (Apr. 2017), pp. 87–154.

[47] F. Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Birkhäuser, Cham, 2015.

[48] T. Sterling and J. J. Irwin. "ZINC 15–ligand discovery for everyone". In: *Journal of chemical information and modeling* 55.11 (2015), pp. 2324–2337.

[49] J. M. Stokes et al. "A Deep Learning Approach to Antibiotic Discovery". In: *Cell* 180.4 (Feb. 2020), 688–702.e13.

[50] S. Ubaru et al. "Fast Estimation of $tr(f(A))$ via Stochastic Lanczos Quadrature". In: *SIAM Journal on Matrix Analysis and Applications* 38.4 (2017), pp. 1075–1099.

[51] D. Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *Journal of chemical information and computer sciences* 28.1 (1988), pp. 31–36.

[52] Y. B. Zel'dovich and A. S. Kompaneetz. "Towards a theory of heat conduction with thermal conductivity depending on the temperature". In: *Collection of papers dedicated to 70th birthday of Academician AF Ioffe, Izd. Akad. Nauk SSSR, Moscow* (1950), pp. 61–71.

A Proofs

A.1 Proof of Lemma 3.1

Starting from the original form of the potential energy functional $\mathcal{V}$ in Equation (3), and using the expression $\rho = (\nabla u)_\sharp \rho_t$ we have:

$$\mathcal{V}(\rho) = \mathcal{V}\big((\nabla u)_\sharp \rho_t\big) = \int V(x)\,\mathrm{d}\left[(\nabla u)_\sharp \rho_t\right] = \int (V \circ \nabla u)\,\mathrm{d}\rho_t \tag{17}$$

On the other hand, for an interaction functional $\mathcal{W}$ we first note that it can be written as

$$\mathcal{W}(\rho) = \frac{1}{2}\iint W(x-x')\,\mathrm{d}\rho(x')\,\mathrm{d}\rho(x) = \frac{1}{2}\int (W * \rho)(x)\,\mathrm{d}\rho(x). \tag{18}$$

In addition, we will need the fact that

$$W * [(\nabla u)_\sharp \rho] = \int W(x-y)\,\mathrm{d}\left[(\nabla u)_\sharp \rho(y)\right] = \int W(x - \nabla u(y))\,\mathrm{d}\rho(y). \tag{19}$$

Hence, combining the two equations above we have:

$$\begin{aligned}
\mathcal{W}\big((\nabla u)_\sharp \rho_t\big) &= \frac{1}{2}\int (W * (\nabla u)_\sharp \rho_t)(x)\,\mathrm{d}\left[(\nabla u)_\sharp \rho_t(x)\right] \\
&= \frac{1}{2}\int \left(\int W(x - \nabla u(y))\,\mathrm{d}\rho_t(y)\right)\mathrm{d}\left[(\nabla u)_\sharp \rho_t(x)\right] \\
&= \frac{1}{2}\iint W(\nabla u(x) - \nabla u(y))\,\mathrm{d}\rho_t(y)\,\mathrm{d}\rho_t(x),
\end{aligned}$$

as stated. □

A.2 Proof of Lemma 3.2

Following Santambrogio [46], we note that whenever u is convex and ν is absolutely continuous, then $\rho = T_\sharp \nu$ is absolutely continuous too, with a density given by

$$\rho = \frac{\nu}{|\mathbf{J}_T|} \circ T^{-1} \tag{20}$$

where $\mathbf{J}_T$ is the Jacobian matrix of T. In our case $\rho = (\nabla u)_\sharp \rho_t$, so that

$$\rho(y) = \left[\frac{\rho_t}{|\mathbf{H}_u|} \circ (\nabla u)^{-1}\right](y) = \frac{\rho_t\left((\nabla u)^{-1}(y)\right)}{\left|\mathbf{H}_u\left((\nabla u)^{-1}(y)\right)\right|} \tag{21}$$

where $\mathbf{H}$ is the Hessian of u. When u is strictly convex it is known that it is invertible and that $(\nabla u)^{-1} = \nabla u^*$, where u^* is the convex conjugate of u (see e.g. Rockafellar [44]). □

A.3 Proof of Corollary 3.3

In this proof, we drop the index τ. As before, we use the change of variables $\rho_t = (\nabla u_t)_\sharp \rho_{t-1}$. Thus, by induction,

$$\rho_t = (\nabla u_t \circ \nabla u_t \cdots \circ \nabla u_1)_\sharp \rho_0 \tag{22}$$

Let $T_{1:t} = (\nabla u_t \circ \nabla u_t \cdots \circ \nabla u_1)$, so that $\rho_t = (T_{1:t})_\sharp \rho_0 = (\nabla u_t \circ T_{1:t-1})_\sharp \rho_0$. The Jacobian of this map is given by the chain rule as:

$$\mathbf{J}_{T_{1:t}}(x) = \mathbf{H}_{u_t}(T_{1:t-1}(x))\mathbf{J}_{T_{1:t-1}}(x) \tag{23}$$

Hence by induction we have:

$$\mathbf{J}_{T_{1:t}}(x) = \Pi_{s=1}^{t}\mathbf{H}_{u_s}(T_{1:s-1}(x)) \tag{24}$$

On the other hand, as long as the inverses exist, we have:

$$T_{1:t}^{-1} \quad = \quad (\nabla u_t)^{-1} \circ \cdots \circ (\nabla u_1)^{-1} = \nabla u_t^* \circ \cdots \circ \nabla u_1^*$$

Hence,

$$\rho_t(y) = \left(\frac{\rho_0}{|\mathbf{J}_{T_{1:t}}|} \circ T_{1:t}^{-1}\right)(y) = \left(\frac{\rho_0}{\Pi_{s=1}^t |\mathbf{H}_{u_s}(T_{1:s-1})|} \circ T_{1:t}^{-1}\right)(y)$$
$$= \frac{\rho_0\left(T_{1:t}^{-1}(y)\right)}{\Pi_{s=1}^t |\mathbf{H}_{u_s}(T_{1:s-1} \circ T_{1:t}^{-1}(y))|},$$

and finally taking the log we obtain:

$$\log(\rho_t) = \left(\log(\rho_0) - \sum_{s=1}^{t} \log(|\mathbf{H}_{u_s}(T_{1:s-1})|)\right) \circ (T_{1:t})^{-1}. \qquad \square \tag{25}$$

A.4 Proof of Lemma 3.4

As before, let $\rho_{t+1} = (\nabla u_{t+1})_\sharp \rho_t$. Thus, by induction,

$$\rho_{t+1} = (\nabla u_{t+1} \circ \nabla u_t \cdots \circ \nabla u_1)_\sharp \rho_0 \tag{26}$$

Let $T_{1:t} = (\nabla u_t \circ \nabla u_t \cdots \circ \nabla u_1)$, so that $\rho_{t+1} = (T_{1:t+1})_\sharp \rho_0 = (\nabla u_{t+1} \circ T_{1:t})_\sharp \rho_0$. The Jacobian of this map is given by the chain rule as:

$$\mathbf{J}_{T_{1:t+1}}(x) = \mathbf{H}_{u_{t+1}}(T_{1:t}(x)) \mathbf{J}_{T_{1:t}}(x) \tag{27}$$

On the other hand, using the fact that (whenever the inverses exist) $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$, in our case we have

$$T_{1:t+1}^{-1} = T_{1:t}^{-1} \circ \nabla u_{t+1}^{-1}, \tag{28}$$

Using (26) and (20), we can write the density of ρ_{t+1} as

$$\rho_{t+1}(y) = \left(\frac{\rho_0}{|\mathbf{J}_{T_{1:t+1}}|} \circ T_{1:t+1}^{-1}\right)(y) \tag{29}$$

$$= \frac{\rho_0(T_{1:t+1}^{-1}(y))}{|\mathbf{J}_{T_{1:t+1}}(T_{1:t+1}^{-1}(y))|} = \frac{\rho_0\left(T_{1:t}^{-1} \circ \nabla u_{t+1}^{-1}(y)\right)}{|\mathbf{H}_{u_{t+1}}(\nabla u_{t+1}^{-1}(y))||\mathbf{J}_{T_{1:t}}(T_{1:t}^{-1} \circ \nabla^{-1} u_{t+1}(y))|} \tag{30}$$

Finally, using the change of variables $y = \nabla u_{t+1} \circ T_{1:t}(x), x = T_{1:t}^{-1} \circ \nabla u_{t+1}^{-1}(y)$, in the integral in the definition of $\mathcal{F}$, we get

$$\mathcal{F}(\rho_{t+1}) = \int f\left(\frac{\rho_0(x)}{|\mathbf{H}_{u_{t+1}}(T_{1:t}(x))||\mathbf{J}_{T_{1:t}}(x)|}\right) |\mathbf{H}_{u_{t+1}}(T_{1:t}(x))||\mathbf{J}_{T_{1:t}}(x)| \, \mathrm{d}x$$
$$= \mathbb{E}_{x \sim \rho_0}\left[f\left(\frac{\rho_0(x)}{|\mathbf{H}_{u_{t+1}}(T_{1:t}(x))||\mathbf{J}_{T_{1:t}}(x)|}\right) \frac{|\mathbf{H}_{u_{t+1}}(T_{1:t}(x))||\mathbf{J}_{T_{1:t}}(x)|}{\rho_0(x)}\right]. \qquad \square$$

B Practical Considerations

B.1 Input Convex Neural Networks

Input Convex Neural Networks were introduced by Amos et al. [2]. A k-layer fully input convex neural network (FICNN) is one in which each layer has the form:

$$z_{i+1} = g_i\left(W_i^{(z)} z_i + W_i^{(y)} y + b_i\right) \quad i = 0, \ldots, k-1 \tag{31}$$

where g_i are activation functions. Amos et al. [2] showed that the function $f : x \mapsto z_k$ is convex with respect to x if all the $W_{i:k-1}^{(z)}$ are non-negative, and all the activation functions g_i are convex and non-decreasing. Residual skip connections from the input with linear weights are also allowed and preserve convexity [2, 26].

In our experiments, we parametrize the Brenier potential u_θ as a FICNN with two hidden layers, with $(100, 20)$ hidden units for the simple PDE experiments in Section 5 and $(100, 100)$ for the molecule

generation experiments in Section 6. In order to preserve the convexity of the network, we clip the weights of $W_{i:k-1}^{(z)}$ after every gradient update using $w_{ij} \leftarrow \max\{w_{ij}, 10^{-8}\}$. Alternatively, one can add a small term $+\lambda\|x\|_2^2$ to enforce strong convexity. In all our simple PDE experiments (§5), we use the ADAM optimizer with 10^{-3} initial learning rate, and a JKO step-size $\tau = 10^{-3}$. Optimization details for the molecular experiments are provided in that section (§6).

B.2 Surrogate loss for entropy

For the choice $f(t) = t \log t$ in the internal energy functional $\mathcal{F}$, we do not use Lemma 3.4 but rather derive the expression from first principles:

$$\begin{aligned}\mathcal{F}\big((\nabla_x u_\theta)_\sharp \rho_t\big) &= \int \frac{\rho_t(x)}{|\mathbf{H}_{u_\theta}(x)|} \log \frac{\rho_t(x)}{|\mathbf{H}_{u_\theta}(x)|} |\mathbf{H}_{u_\theta}(x)| \,\mathrm{d}x \\ &= \int \log \frac{\rho_t(x)}{|\mathbf{H}_{u_\theta}(x)|} \rho_t(x) \,\mathrm{d}x \\ &= \int \rho_t(x) \log \rho_t(x) \,\mathrm{d}x - \int \log |\mathbf{H}_{u_\theta}(x)| \rho_t(x) \,\mathrm{d}x = \mathcal{F}(\rho_t) - \mathop{\mathbb{E}}_{x\sim\rho_t}[\log |\mathbf{H}_{u_\theta}(x)|]\end{aligned}$$

As mentioned earlier, this expression has an interesting interpretation as reducing negative entropy (increasing entropy) of ρ_t by an amount given by a log-determinant barrier term on u's Hessian. We see that the only term depending on θ is

$$- \mathop{\mathbb{E}}_{x\sim\rho_t}[\log |\mathbf{H}_{u_\theta}(x)|].$$

We discuss how to estimate this quantity and backpropagate through $\mathbf{H}_{u_\theta}$ in Section B.4.

B.3 Surrogate losses for internal energies

Let

$$r_x(u_\theta) = \log(\xi(x)) = \log(\rho_0(x)) - \log |H_{u_\theta}(T_{1:t}(x))| - \log(|J_{1:T}(x)|)$$

From Lemma 3.4, our point-wise loss is :

$$L(u_\theta) = \frac{f \circ \exp(r_x(u_\theta))}{\exp(r_x(u_\theta))}$$

Computing gradient w.r.t θ_i parameters of u_θ:

$$\begin{aligned}\frac{\partial}{\partial \theta_i} L(u) &= \frac{f'(\exp(r_x(u_\theta)))[\exp(r_x(u_\theta))]^2 \frac{\partial}{\partial \theta_i} r_x(u_\theta) - \exp(r_x(u_\theta)) \frac{\partial}{\partial \theta_i} r_x(u_\theta) f \circ \exp(r_x(u_\theta))}{[\exp(r_x(u_\theta)]^2} \\ &= \left(\frac{f'(\exp(r_x(u_\theta))) \exp(r_x(u_\theta)) - f(\exp(r_x(u_\theta)))}{\exp(r_x(u_\theta))} \right) \frac{\partial}{\partial \theta_i} r_x(u_\theta)\end{aligned}$$

Also,

$$\frac{\partial}{\partial \theta_i} r_x(u) = -\frac{\partial}{\partial \theta_i} \log |H_{u_\theta}(T_{1:t}(x))|$$

Hence the Surrogate loss that has same gradient can be evaluated as follows:

$$\mathcal{L}(u_\theta) = -\underbrace{\left(\frac{f'(\exp(r_x(u_\theta))) \exp(r_x(u_\theta)) - f(\exp(r_x(u_\theta)))}{\exp(r_x(u_\theta))} \right)}_{\text{no grad}} \log |H_{u_\theta}(T_{1:t}(x))|$$

For the particular case of porous medium internal energy, let $a = \exp(r)$. For $f(a) = \frac{1}{m-1} a^m = \frac{1}{m-1} \exp(m \log(a)) = \frac{1}{m-1} \exp(mr))$, we have $f'(a) = \frac{m}{m-1} a^{m-1} = \frac{m}{m-1} \exp((m-1)\log(a))$ $f'(a) = \frac{m}{m-1} \exp((m-1)r)$.

$$f'(a) - \frac{f(a)}{a} = \frac{m}{m-1} \exp((m-1)r) - \frac{1}{m-1} \exp((m-1)r) = \exp((m-1)r)$$

Hence we have finally the surrogate loss:

$$\mathcal{L}(u_\theta) = -\underbrace{\exp((m-1)r_x(u_\theta))}_{\text{no grad}} \log |H_{u_\theta}(T_{1:t}(x))|,$$

for which we discuss estimation in Section B.4. Table 3 summarizes surrogates losses for common energies in gradient flows.

Table 3: Surrogate optimization objectives used for computation. Here $\hat{\mathbb{E}}$ denotes an empirical expectation, $\xi(x)$ is defined in Lemma 3.4, and SG denotes the STOPGRAD operator.

Functional Type	Exact Form $F(\rho)$	Surrogate Objective $\hat{F}(u_\theta)$
Potential energy	$\int V(x)\,\mathrm{d}\rho(x)$	$\hat{\mathbb{E}}_{x\sim\rho_t} V(\nabla_x u_\theta(x))$
Interaction energy	$\iint W(x-x')\,\mathrm{d}\rho(x)\,\mathrm{d}\rho(x')$	$\frac{1}{2}\hat{\mathbb{E}}_{x,y\sim\rho_t} W(\nabla_x u_\theta(x) - \nabla_x u_\theta(y))$
Neg-Entropy	$\int \rho(x)\log\rho(x)\,\mathrm{d}x$	$-\hat{\mathbb{E}}_{x\sim\rho_t} \log\lvert\mathbf{H}_{u_\theta}(x)\rvert$
Nonlinear diffusion	$\int \frac{1}{m-1}\rho(x)^m\,\mathrm{d}x$	$-\hat{\mathbb{E}}_{x\sim\rho_0} e^{(m-1)\mathrm{SG}(\log\xi(x))} \log\lvert\mathbf{H}_{u_\theta}(T_{1:t}(x))\rvert$

B.4 Stochastic log determinant estimators

For numerical reasons, we use different methods to evaluate and compute gradients of Hessian log-determinants.

Evaluating log-determinants Following [26] we use Stochastic Lanczos Quadrature (SLQ) [50] to estimate the log determinants. We refer to this estimation step as LOGDETESTIMATOR.

Estimating gradients of log-determinants The SLQ procedure involves an eigendecomposition, which is unstable to back-propagate through. Thus, to compute gradients, Huang et al. [26], inspired by Chen et al. [14], instead use the following expression of the Hessian log-determinant:

$$\frac{\partial}{\partial\theta}\log|\mathbf{H}| = \frac{1}{|\mathbf{H}|}\frac{\partial}{\partial\theta}|\mathbf{H}| = \frac{1}{|\mathbf{H}|}\mathrm{tr}(\mathrm{adj}(\mathbf{H})\frac{\partial H}{\partial\theta}) = \mathrm{tr}(\mathbf{H}^{-1}\frac{\partial H}{\partial\theta}) = \mathop{\mathbb{E}}_{v}\big[v^\top\mathbf{H}^{-1}\frac{\partial\mathbf{H}}{\partial\theta}v\big], \tag{32}$$

where v is a random Rademacher vector. This last step is the Hutchinson trace estimator [27].

As Huang et al. [26], we avoid constructing and inverting the Hessian in this expression by instead solving a problem that requires computing only Hessian-vector products:

$$\operatorname*{argmin}_z \tfrac{1}{2}z^\top\mathbf{H}z - v^\top z \tag{33}$$

Since $\mathbf{H}$ is symmetric positive definite, this strictly convex problem has a unique minimizer, z^*, that satisfies $z^* = \mathbf{H}^{-1}v$. This problem can be solved using the conjugate gradient method with a fixed number of iterations or a error stopping condition. Thus, computing the last expression in Eq. (32) can be done with automatic differentiation by: (i) sampling a Rademacher vector v, (ii) running conjugate gradient for m iterations on problem (33) to obtain z^m, (iii) computing $\frac{\partial}{\partial\theta}[(z^m)^\top\mathbf{H}v]$ with automatic differentiation.

Algorithm 2: Density estimation for JKO-ICNN

Input: Query point x, sequence of Brenier potentials $\{u_i\}_{i=0}^T$ obtained with JKO-ICNN, initial density evaluation function $\rho_0(\cdot)$.
Initialize $y_t \leftarrow x$
for $t = T$ **to** 1 **do**
  $y_{t-1} \leftarrow \mathrm{argmax}_y \langle y_t, y\rangle - u_t(y)$
  $\{y_{t-1}$ satisfies $(\nabla u_t)(y_{t-1}) = y_t\}$
end for
$x_0 \leftarrow y_0$
for $t = 0$ **to** $T-1$ **do**
  $x_{t+1} \leftarrow \nabla u_t(x_t)$
end for
{Compute $\delta \triangleq \log|\mathbf{J}_{T_{1:t}}(x_0)|$}
$\delta \leftarrow$ LOGDETESTIMATOR(x_0, x_T)
$p \leftarrow \log\rho_0(y_0) - \delta$
Output: p satisfying $p = \log\rho_t(x)$

B.5 Implementation Details

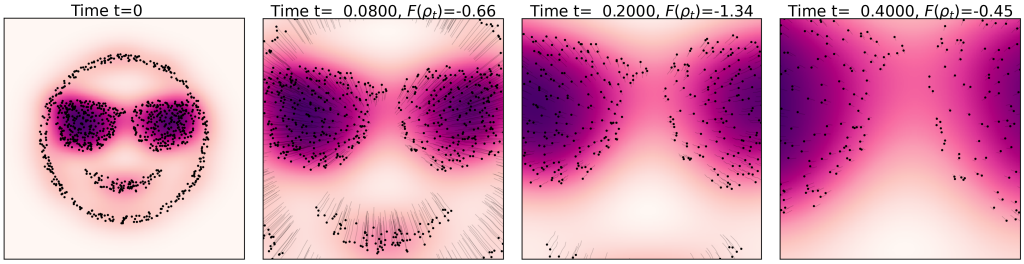
Apart from the stochastic log-determinant estimation (Section B.4) needed for computing internal energy functionals, the other main procedure that requires discussion is the density estimation. This is needed, for example, to obtain exact evaluation of the internal energy functionals $\mathcal{F}$ and requires having access to the exact density (or an estimate thereof, e.g., via KDE) from which the initial set of particles were sampled. For this, we rely on Lemma 3.2 and Corollary 3.3, which combined provide a way to estimate the density $\rho_T(x)$ using $\rho_0(x)$, the sequence of Brenier potentials $\{u_i\}_{i=1}^T$, and their combined Hessian log-determinant. This procedure is summarized in Algorithm 2.

B.6 Assets

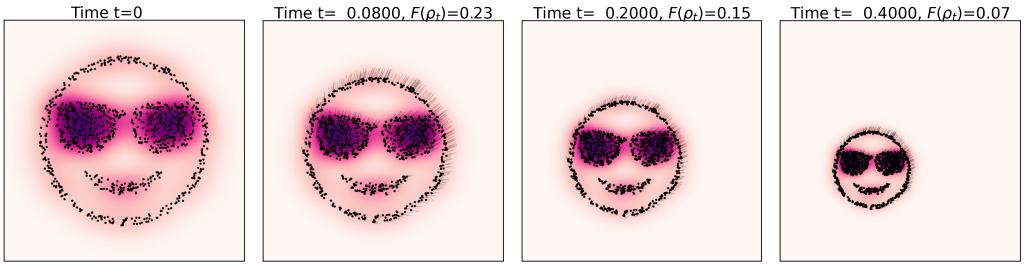
Software Our implementation of JKO-ICNN relies on various open-source libraries, including pytorch [38] (license: BSD), pytorch-lightning [18] (Apache 2.0), POT [20] (MIT), geomloss [19] (MIT), rdkit [31, 32] (BSD 3-Clause).

Data All data used in Section 5 is synthetic. The MOSES dataset is released under the MIT license. The QM9 dataset [42, 45] does not explicitly provide a license in their website nor data files.

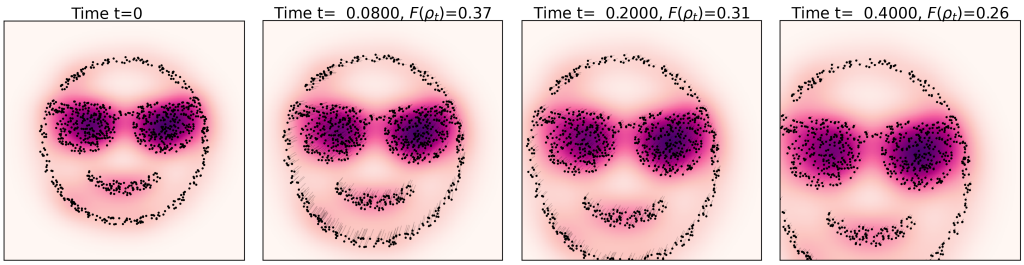
C Additional qualitative results on 2D datasets



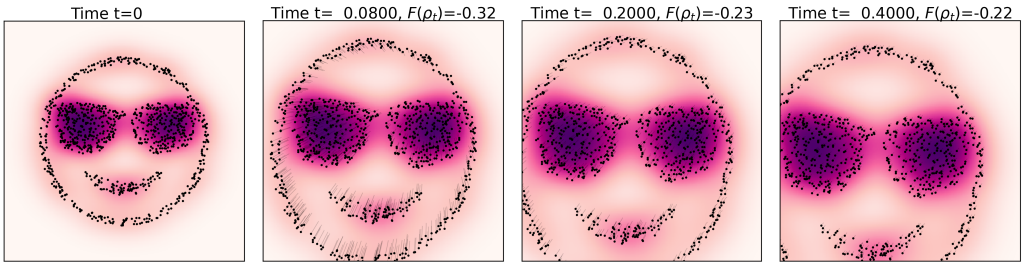
(a) Heat equation corresponding to internal energy functional $\mathcal{F}(\rho) = \int \rho(x) \log \rho(t) \, \mathrm{d}x$ (see Table 1).



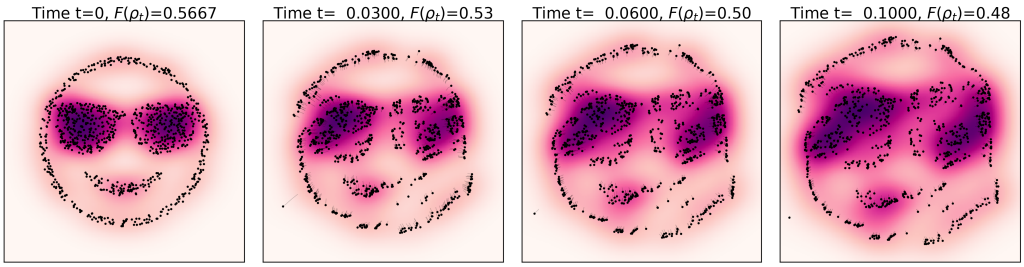
(b) Potential-only flow using functional $\mathcal{V}(\rho) = \int \|x - x_0\|_2^2 \, \mathrm{d}\rho(x)$.



(c) Fokker-Planck with linear diffusion: same potential as 4b, plus $\mathcal{F}(\rho) = \int \rho(x) \log \rho(x) \, \mathrm{d}x$.



(d) Fokker-Planck with nonlinear diffusion: same potential as 4b, plus $\mathcal{F}(\rho) = \int \rho^2(x) \, \mathrm{d}x$.



(e) Aggregation equation (same functional as in §5.3).

Figure 4: JKO-ICNN flows of a 2D point cloud with density estimated via KDE.

C.1 Experimental details: PDEs with known solutions

In all the experiments in Section 6, we use the same JKO step-size $\tau = 10^{-3}$ for the outer loop and an ADAM optimizer with initial learning rate $\eta = 10^{-3}$ for the inner loop. We run the inner optimization loop for 400 iterations. In the plots in Figure 1, we show snap-shots at different intervals to facilitate visualization. For these experiments, we parametrize u_θ as an 2-hidden-layer ICNN with layer width: $(100, 20)$. For the non-linear diffusion term needed for the PDEs in Sections 5.1 and 5.2, we use the surrogate objective (14). In all cases, we impose strict positivity on the weight matrices $W^{(z)}$ (Eq. (31)) with minimum value $\delta = 10^{-18}$ to enforce strong convexity.

D Experimental details: molecular discovery with JKO-ICNN

In what follows, we present the experimental details of the experiments in Section 6, on the MOSES dataset. The MOSES dataset [41], which is a subset of the ZINC database [48]. MOSES dataset is available for download at https://github.com/molecularsets/moses and is released under the MIT license.

D.1 Experimental pipeline

All Molecular discovery JKO-ICNN experiments were run in a compute environment with 1 CPU and 1 V100 GPU submitted as resource-restricted jobs to a cluster. This applies to both convex QED surrogate classifier training and evaluation runs and to the JKO-ICNN flows for each configuration of hyperparameters and random seed initialization.

The pipeline and results presented in this subsection are based on the MOSES dataset [41].

Convex QED surrogate classifier We first describe the hyperparameters and results of the convex surrogate that was trained to predict high (> 0.85) and low (< 0.85) QED values from molecule embeddings coming from a pre-trained VAE. Molecules with high QED were given lower value labels compared to the low QED molecules so that when this model would be used as potential, minimizing this functional would lead to higher QED values. For this convex surrogate, we trained a Residual ICNN [2, 26] model with four hidden layers, each with dimension 128, which was the dimensionality of the input molecule embeddings as well. We trained the model with binary cross-entropy loss. To maintain convexity in the potential functional however, we used the last layer before the sigmoid activation for V. The model was trained with an initial learning rate of 0.01, batch sizes of 1,024, ADAM optimizer, and a learning rate scheduler that decreased learning rate on validation set loss plateau. The model was trained for 100 epochs, and the weights from the final epoch were used to initialize the convex surrogate in the potential functional. For this epoch, the model achieved 85% accuracy on the test set. In Figure 5, we display the test set confusion matrix for this final epoch.

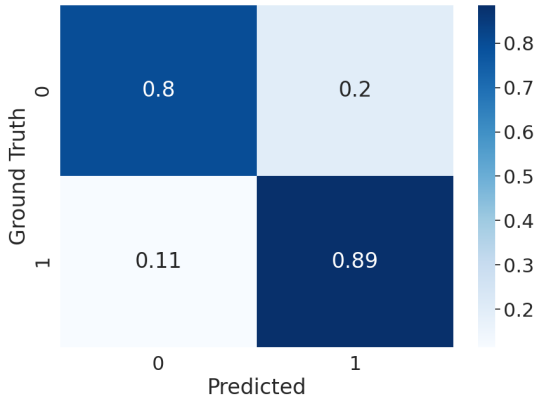


Figure 5: Confusion matrix for the convex surrogate trained to predict QED labels from molecule embeddings for MOSES.

JKO-ICNN QED histogram values In Table 4, we present the same results as in Section 6 but include mean and standard deviation QED values for the point clouds.

Table 4: Molecular discovery with JKO-ICNN experiment results: Measures of validity, uniqueness, and median, average, and standard deviation QED are reported in each row for the corresponding point cloud of embeddings. For each point cloud, we decode the embeddings to get SMILES strings and use RDKit to determine whether the corresponding string is valid and get the associated QED value. In the first row, we display the point cloud at time step zero of JKO-ICNN. In the subsequent rows, we display the values for each measurement at the final time step T of JKO-ICNN for different hyperparameter configurations of distribution distance D (either $\mathrm{Sinkhorn}$ or MMD) and weight on this distance λ_2 (either 1,000 or 10,000). Each measurement value cell contains mean values $\pm$ one standard deviation for five repeated runs of the experiment with different random initialization seeds. We find that the setup that uses $\mathrm{Sinkhorn}$ with weight 10,000 yields the best results in terms of moving the point cloud towards regions with higher QED without sacrificing validity and uniqueness of the decoded SMILES strings. This table contains results for the MOSES dataset

Measure	D	λ_2	Validity	Uniqueness	QED Median	QED Avg.	QED Std.
ρ_0	N/A	N/A	100.000 ± 0.000	99.980 ± 0.045	0.630 ± 0.001	0.621 ± 0.000	0.063 ± 0.002
ρ_T^τ	$\mathrm{Sinkhorn}$	1,000	92.460 ± 2.096	69.919 ± 4.906	0.746 ± 0.016	0.735 ± 0.009	0.110 ± 0.003
ρ_T^τ	$\mathrm{Sinkhorn}$	10,000	93.020 ± 1.001	99.245 ± 0.439	0.769 ± 0.002	0.754 ± 0.003	0.112 ± 0.002
ρ_T^τ	MMD	1,000	94.560 ± 1.372	51.668 ± 2.205	0.780 ± 0.009	0.767 ± 0.013	0.107 ± 0.012
ρ_T^τ	MMD	10,000	92.020 ± 3.535	53.774 ± 3.013	0.776 ± 0.014	0.767 ± 0.009	0.102 ± 0.011

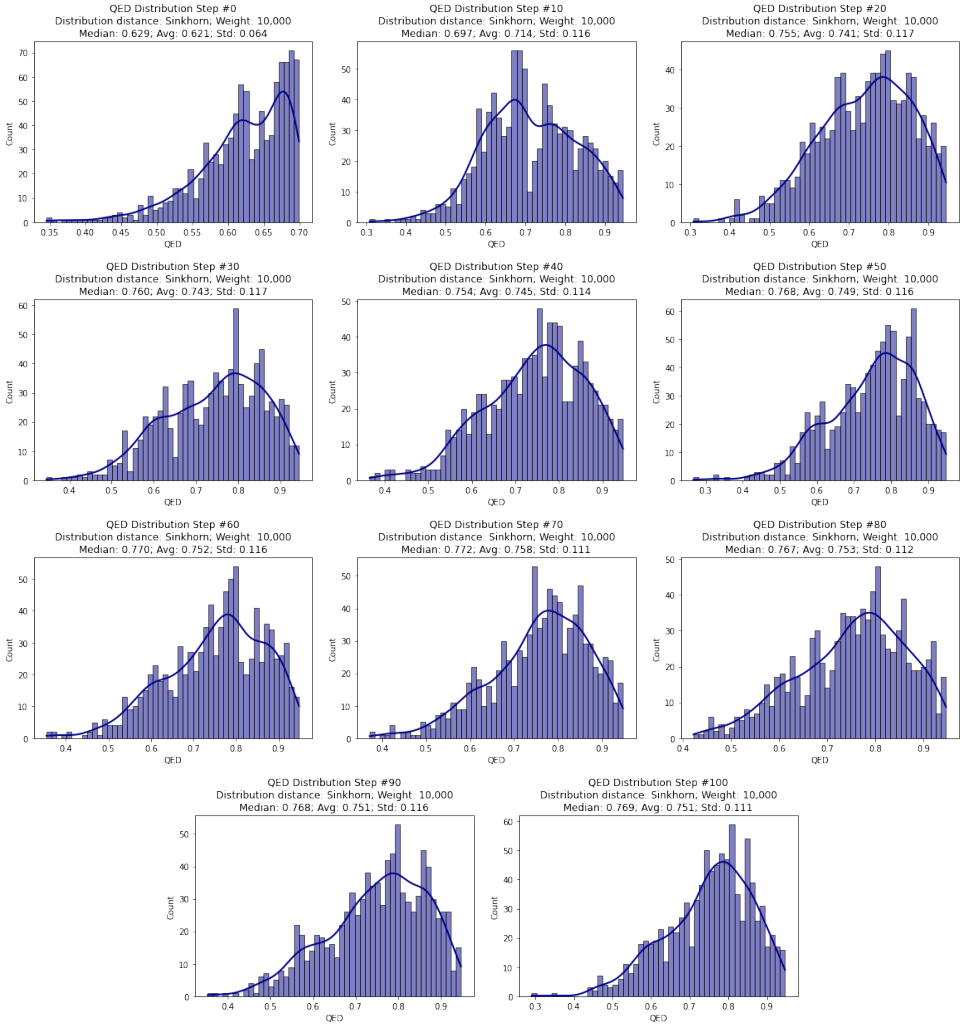


Figure 6: Histograms of QED values for the decoded SMILES strings corresponding to the point clouds at time step $t = 0, 10, 20 \ldots 100$ for the JKO-ICNN experiment that uses $\mathrm{Sinkhorn}$ as the distribution distance with weight 10,000. We observe a clear shift to the right in the distribution, which corresponds to increased drug-likeness of the decoded molecule strings. This figure displays results for the MOSES dataset experiment.

JKO-ICNN QED Histogram Trajectories In Figure 6, we present several time steps of the histograms of the QED values for the decoded SMILES strings corresponding to the point clouds for the experiment that used Sinkhorn as the distribution distance with weight 10,000.

E Molecule JKO-ICNN experimental setup and results for QM9 dataset

In this section, we repeat the results and analysis presentation of D.1 but for the experiments that used the QM9 dataset.

The QM9 dataset contains on average smaller molecules than MOSES. The MOSES dataset is larger in size than QM9 and contains about 1.6M training (compared to 121k in QM9) and 176k test molecules (compared to 13k in QM9).

For these experiments a separate VAE model and convex surrogate classifier were trained using the QM9 dataset, which was split into about 121k training and 13k test molecules.

Convex QED surrogate classifier (QM9) For QM9, the convex surrogate was trained to predict high (> 0.5) and low (< 0.5) QED values from molecule embeddings coming from a pre-trained VAE. The threshold for QM9 molecules was set to a lower value compared to that for the MOSES dataset because the underlying QED distribution of train and test data for QM9 molecules has significantly lower values compared to MOSES. As above, molecules with high QED were given lower value labels compared to the low QED molecules so that when this model would be used as potential, minimizing this functional would lead to higher QED values. Similar to the MOSES dataset pipeline, for this convex surrogate, we trained a Residual ICNN [2, 26] model with four hidden layers, each with dimension 128, which was the dimensionality of the input molecule embeddings as well. We trained the model with binary cross-entropy loss. To maintain convexity in the potential functional, we used the last layer before the sigmoid activation for V. The model was trained with an initial learning rate of 0.01, batch sizes of 1,024, ADAM optimizer, and a learning rate scheduler that decreased learning rate on validation set loss plateau. The model was trained for 100 epochs, and the weights from the final epoch were used to initialize the convex surrogate in the potential functional. For this epoch, the model achieved 88.6% accuracy on the test set. In Figure 7, we display the test set confusion matrix for this final epoch.

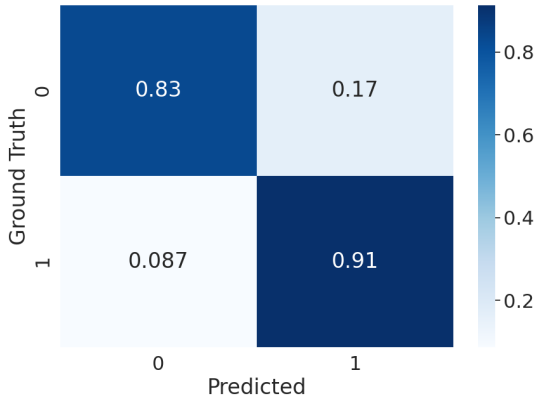


Figure 7: Confusion matrix for the convex surrogate trained to predict QED labels from molecule embeddings for QM9.

JKO-ICNN QED Histogram values (QM9) In Table 5, we present the same results as in Appendix D.1, which includes mean and standard deviation QED values for the point clouds for the experiment that used the QM9 dataset.

For the JKO-ICNN flow on the QM9 dataset, we started with an initial distribution of embeddings ρ_0 that had corresponding QED value of less than 0.35. This initial point cloud was taken from the QM9 train set since the test set is quite small and does not contain enough data points below the starting QED threshold.

Table 5: Molecular discovery with JKO-ICNN experiment results: Measures of validity, uniqueness, and median, average, and standard deviation QED are reported in each row for the corresponding point cloud of embeddings. For each point cloud, we decode the embeddings to get SMILES strings and use RDKit to determine whether the corresponding string is valid and get the associated QED value. In the first row, we display the point cloud at time step zero of JKO-ICNN. In the subsequent rows, we display the values for each measurement at the final time step T of JKO-ICNN for different hyperparameter configurations of distribution distance D (either $\mathrm{Sinkhorn}$ or MMD) and weight on this distance λ_2 (either 1,000 or 10,000). Each measurement value cell contains mean values $\pm$ one standard deviation for five repeated runs of the experiment with different random initialization seeds. This table contains results for the QM9 dataset.

Measure	D	λ_2	Validity	Uniqueness	QED Median	QED Avg.	QED Std.
ρ_0	N/A	N/A	100.000 $\pm$ 0.000	99.840 $\pm$ 0.134	0.315 $\pm$ 0.001	0.303 $\pm$ 0.001	0.041 $\pm$ 0.001
ρ_T^τ	$\mathrm{Sinkhorn}$	1,000	90.840 $\pm$ 1.457	33.367 $\pm$ 2.491	0.381 $\pm$ 0.024	0.373 $\pm$ 0.016	0.096 $\pm$ 0.005
ρ_T^τ	$\mathrm{Sinkhorn}$	10,000	92.700 $\pm$ 0.828	81.925 $\pm$ 1.982	0.419 $\pm$ 0.005	0.404 $\pm$ 0.005	0.096 $\pm$ 0.004
ρ_T^τ	MMD	1,000	91.680 $\pm$ 4.463	22.424 $\pm$ 1.185	0.452 $\pm$ 0.024	0.434 $\pm$ 0.019	0.094 $\pm$ 0.005
ρ_T^τ	MMD	10,000.	88.800 $\pm$ 4.661	28.664 $\pm$ 1.500	0.448 $\pm$ 0.013	0.432 $\pm$ 0.010	0.093 $\pm$ 0.005

As seen with the experiment on MOSES, all four combinations are able to increase QED values. However, the setups that use MMD or $\lambda_2 = 1{,}000$ lead to mode collapse, see the discussion in Section 6.

JKO-ICNN QED Histogram Trajectories (QM9) In Figure 8, we present several time steps of the histograms of the QED values for the decoded SMILES strings corresponding to the point clouds for the experiment that used $\mathrm{Sinkhorn}$ as the distribution distance with weight 10,000.

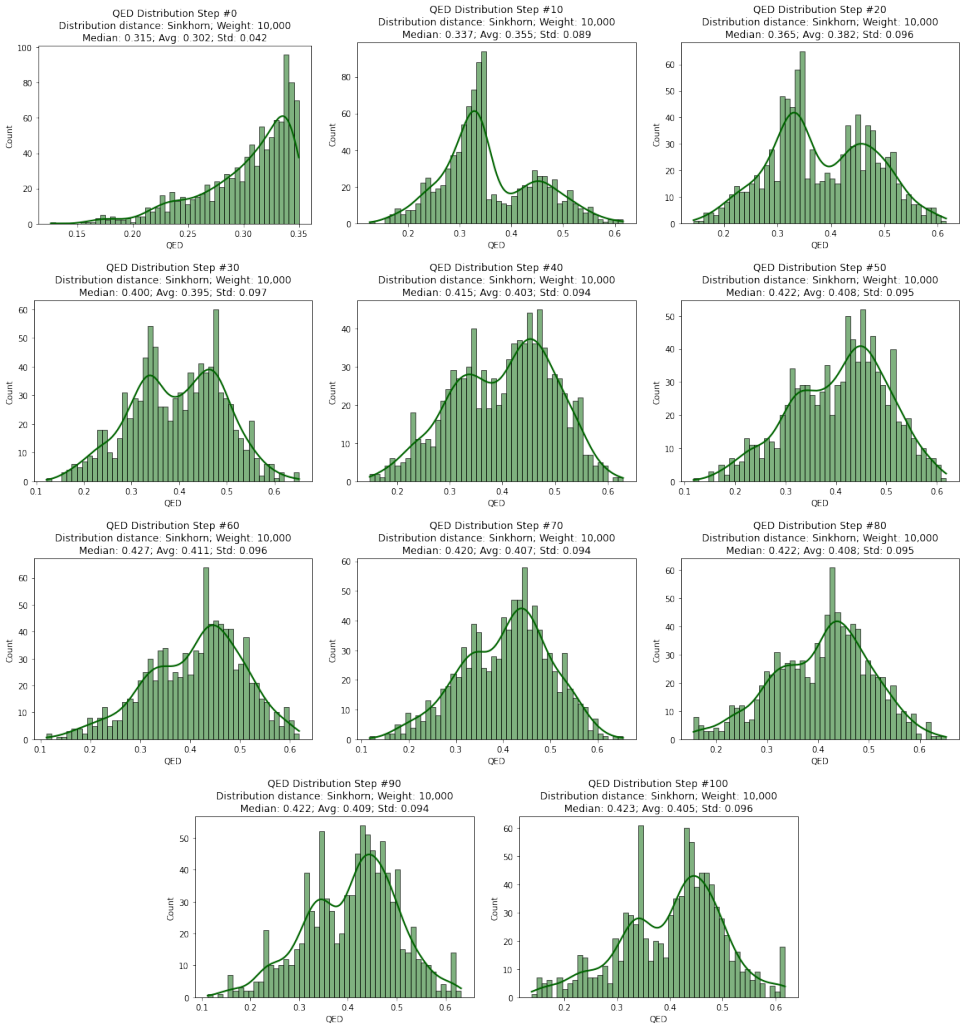


Figure 8: Histograms of QED values for the decoded SMILES strings corresponding to the point clouds at time step $t = 0, 10, 20...100$ for the JKO-ICNN experiment that uses $\mathrm{Sinkhorn}$ as the distribution distance with weight 10,000. We observe a clear shift to the right in the distribution, which corresponds to increased drug-likeness of the decoded molecule strings. This figure displays results for the QM9 dataset experiment.