A machine learning framework for solving high-dimensional mean field game and mean field control problems

Lars Ruthotto[a,b,1], Stanley J. Osher[c,1], Wuchen Li[c], Levon Nurbekyan[c], and Samy Wu Fung[c]

[a]Department of Mathematics, Emory University, Atlanta, GA 30322; [b]Department of Computer Science, Emory University, Atlanta, GA 30322; and [c]Department of Mathematics, University of California, Los Angeles, CA 90095

Contributed by Stanley J. Osher, March 2, 2020 (sent for review December 18, 2019; reviewed by Weinan E and George Em Karniadakis)

Mean field games (MFG) and mean field control (MFC) are critical classes of multiagent models for the efficient analysis of massive populations of interacting agents. Their areas of application span topics in economics, finance, game theory, industrial engineering, crowd motion, and more. In this paper, we provide a flexible machine learning framework for the numerical solution of potential MFG and MFC models. State-of-the-art numerical methods for solving such problems utilize spatial discretization that leads to a curse of dimensionality. We approximately solve high-dimensional problems by combining Lagrangian and Eulerian viewpoints and leveraging recent advances from machine learning. More precisely, we work with a Lagrangian formulation of the problem and enforce the underlying Hamilton–Jacobi–Bellman (HJB) equation that is derived from the Eulerian formulation. Finally, a tailored neural network parameterization of the MFG/MFC solution helps us avoid any spatial discretization. Our numerical results include the approximate solution of 100-dimensional instances of optimal transport and crowd motion problems on a standard work station and a validation using a Eulerian solver in two dimensions. These results open the door to much-anticipated applications of MFG and MFC models that are beyond reach with existing numerical methods.

mean field games | mean field control | machine learning | optimal transport | Hamilton–Jacobi–Bellman equations

Mean field games (MFG) (1–5) and mean field control (MFC) (6) allow one to simulate and analyze interactions within large populations of agents. Hence, these models have found widespread use in economics (7–10), finance (11–14), crowd motion (15–18), industrial engineering (19–21), and, more recently, data science (22) and material dynamics (23).

The theoretical properties of MFG and MFC problems have been continuously developed over the last few decades (see, e.g., refs. 24–29). A key observation is that both problems involve a Hamilton–Jacobi–Bellman (HJB) equation that is coupled with a continuity equation. From the solution of this system of partial differential equations (PDEs), each agent can infer the cost of their optimal action, which is why it is commonly called value function. In addition, the agent can obtain their optimal action from the gradient of the value function, which alleviates the need for individual optimization (see refs. 30 and 31 for details). We note that, due to similar conventions in physics, the value function is sometimes also called a potential.

Our framework applies to a common subclass of MFGs, namely, potential MFGs, and MFCs. These problems can be formulated as infinite-dimensional optimal control problems in density space. Interestingly, their optimality conditions coincide with the HJB and continuity equation.

Despite many theoretical advances, the development of numerical methods for solving MFGs, particularly in high-dimensional sample spaces, lags and has not kept pace with growing data and problem sizes. A crucial disadvantage of most existing approaches for solving MFGs or their underlying HJB equations is their reliance on meshes (see refs. 32–42 and references therein). Mesh-based methods are prone to the curse of dimensionality, that is, their computational complexity grows exponentially with spatial dimension (43).

In this paper, we tackle the curse of dimensionality in two steps. First, extending the approach in ref. 44, we solve the continuity equation and compute all other terms involved in the MFG using Lagrangian coordinates. In practice, this requires the computation of characteristic curves and the Jacobian determinant of the induced transformation; both terms can be inferred from the value function. In our examples, the former follows the gradient, and the logarithm of the latter can be approximated by integrating the Laplacian of the value function. Our scheme also allows us to control and enforce the HJB equation along the characteristics. These computations can be performed independently and in parallel for all agents.

Second, we parameterize the value function in space and time using a neural network that is specifically designed for an accurate and efficient Lagrangian scheme. With this design, we can also directly penalize the violation of the HJB equation. Thereby,

Significance

Mean field games (MFG) and mean field control (MFC) play central roles in a variety of scientific disciplines such as physics, economics, and data science. While the mathematical theory of MFGs has matured considerably, the development of numerical methods has not kept pace with growing problem sizes and massive datasets. Since MFGs, in general, do not admit closed-form solutions, effective numerical algorithms are critical. Most existing numerical methods use meshes and thus are prone to the curse of dimensionality. Our framework is mesh-free, since it combines Lagrangian PDE solvers and neural networks. By penalizing violations of the underlying Hamilton–Jacobi–Bellman equation, we increase accuracy and computational efficiency. Transforming MFGs into machine learning problems promises exciting opportunities to advance application and theory.

Author contributions: L.R., S.J.O., W.L., L.N., and S.W.F. designed research; L.R., L.N., and S.W.F. performed research; and L.R., S.J.O., W.L., L.N., and S.W.F. wrote the paper.

Reviewers: W.E., Princeton University; and G.E.K., Brown University.

The authors declare no competing interest.

Data deposition: Prototype implementation written in Julia is available as open-source software under a permissible license. Readers can access all codes, scripts, and produced results at http://github.com/EmoryMLIP/MFGnet.jl.

[1]To whom correspondence may be addressed. Email: sjo@math.ucla.edu or lruthotto@emory.edu.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.1922204117/-/DCSupplemental.

First published April 9, 2020.

we develop a generic framework that transforms a wide range of MFGs into machine learning (ML) problems.

In our numerical experiments, we solve high-dimensional instances of the dynamical optimal transport (OT) problem (45, 46), a prototypical instance of a potential MFG, and an MFG inspired by crowd motion. In OT, one seeks to find the path connecting two given densities that minimizes a kinetic energy that models transport costs. As for other MFGs, there are many relevant high-dimensional instances related to dynamical OT, for example, in ML (47). We validate our scheme using comparisons to a Eulerian method on two-dimensional (2D) instances. Most crucially, we show the accuracy and scalability of our method using OT and MFG instances in up to 100 space dimensions. We also show that penalizing the HJB violations allows us to solve the problem more accurately with less computational effort. Our results for the crowd motion problem also show that our framework is capable of solving potential MFGs and MFCs with nonlinear characteristics.

To enable further theoretical and computational advances as well as applications to real-world problems, we provide our prototype implementation written in Julia (48) as open-source software under a permissible license.

Related Work

Our work lies at the interface of ML, PDEs, and optimal control. Recently, this area has received a lot of attention, rendering a comprehensive review to be beyond the scope of this paper. The idea of solving high-dimensional PDEs and control problems with neural networks has been pioneered by other works (49–51) and has been further investigated by ref. 52. In this section, we put our contributions into context by reviewing recent works that combine concepts from ML, OT, MFGs, and Lagrangian methods for MFG.

OT and ML. Despite tremendous theoretical insight gained into the problem of optimally transporting one density to match another one, its numerical solution remains challenging, particularly when the densities live in spaces of four dimensions or more. In small-dimensional cases, there are many state-of-the-art approaches that effectively compute the global solution (see, e.g., refs. 40, 42, 53, and 54 and the survey in ref. 55). Due to their reliance on Euclidean coordinates, those techniques require spatial discretization, which makes them prone to the curse of dimensionality. An exception is the approach in ref. 56 that uses a generative model for computing the transport plan. This work uses a neural network model for the density and a Lagrangian PDE solver.

An ML task that bears many similarities to OT is variational inference with normalizing flows (47). Roughly speaking, the goal is to transform given samples from a typically unknown distribution such that they are approximately normally distributed. To this end, one trains a neural network-based flow model; hence, the name normalizing flow. The trained flow can be used as a generator to produce new samples from the unknown distribution by reversing the flow, starting with samples drawn from a normal distribution. While the original formulation of the learning problem in normalizing flows does not incorporate transport costs, refs. 57–59 successfully apply concepts from OT to analyze and improve the learning of flows. The approach in ref. 59 is formulated as a point cloud matching problem and estimates the underlying densities using Gaussian mixture models. The works (57, 58) propose neural network models for the value function instead of the velocity of the flow, which leads to more-physically plausible models. This parameterization has also been used to enforce parts of the HJB equation via quadratic penalties (58). We generalize these ideas from OT to a broader class of MFGs that naturally incorporate transport costs. We also add a penalty for the final time condition of the HJB to the training problem.

MFGs and ML. ML and MFGs have become increasingly intertwined in recent years. On the one hand, MFG theory has been used to provide valuable insight into the training problems in deep learning (22). On the other hand, refs. 60–62 use ML to solve MFGs in up to four spatial dimensions. The methods in refs. 61 and 62 are limited to MFGs whose formulations do not involve the population density, as this computation is challenging. For the time-independent second-order problems in ref. 62, one can express the density explicitly in terms of the value function. Furthermore, in numerical examples for the time-dependent case in ref. 61, the congestion terms depend on the average positions of the population. In this situation, the congestion term can be computed using sample averages. Our framework does not have the above limitations and, in particular, is applicable to MFGs where there is no analytical formula for the density or special structure that can be used to compute the congestion term, for example, MFGs with nonlinear congestion terms. We achieve this using the Lagrangian formulation that includes an estimate of the density along the agents' trajectories. This generality is a critical contribution of our work.

Additionally, our neural network architecture for the control respects the structure induced by optimality conditions. We believe that this property is critical for obtaining accurate algorithms that scale and yield correct geometry for the agents' trajectories. As a result, we use our method to approximately solve MFGs in 100 dimensions on a standard workstation.

Lagrangian Methods in MFG. To the best of our knowledge, the first Lagrangian method for solving MFG problems appeared in ref. 44. Lagrangian techniques are natural from an optimal control perspective and unavoidable for high-dimensional problems. However, a crucial computational challenge in applying these techniques in MFG stems from the density estimation, which is critical, for example, to compute the congestion cost incurred by an individual agent. In ref. 44, the authors overcome this difficulty for nonlocal interactions by passing to Fourier coordinates in the congestion term and thus avoiding the density estimation. Our neural network parameterization aims to reduce the computational effort and memory footprint of the methods in ref. 44 and provides a tractable way to estimate the population density.

Lagrangian methods for mass-transport problems in image processing were proposed in ref. 63. While the computation of the characteristics is mesh-free, the final density is computed using a particle-in-cell method that does not scale to high-dimensional problems.

Mathematical Modeling

This section provides a concise mathematical formulation of MFG and MFC models and key references; for more details, see the monographs (3, 6). MFGs model large populations of rational agents that play a noncooperative differential game. At optimality, this leads to a Nash equilibrium where no single agent can do better by unilaterally changing their strategy. By contrast, in MFC, there is a central planner that aims at a distributed strategy for agents to minimize the average cost or maximize the average payoff across the population. Starting with a microscopic description of MFGs, we derive their macroscopic equations, introduce the important class of potential MFGs, and briefly outline MFCs.

MFGs. Assume that a continuum population of small rational agents plays a noncooperative differential game on a time horizon $[0, T]$. Suppose that an agent is positioned at $x \in \mathbb{R}^d$ at time $t \in [0, T]$. For fixed t, we denote agents' population density by $\rho(\cdot, t) \in \mathcal{P}(\mathbb{R}^d)$, where $\mathcal{P}(\mathbb{R}^d)$ is the space of all probability densities. The agent's cost function is given by

$$J_{x,t}(v,\rho)=\int_t^T L(z(s),v(s))+F(z(s),\rho(z(s),s))\,ds \\ +G(z(T),\rho(z(T),T)), \quad [1]$$

where $v:[0,T]\to\mathbb{R}^d$ is the strategy (control) of this agent, and their position changes according to

$$\partial_t z(t)=v(t),\ 0\le t\le T, \quad z(0)=x. \quad [2]$$

In Eq. **1**, $L:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ is a running cost incurred by an agent based solely on their actions, $F:\mathbb{R}^d\times\mathcal{P}(\mathbb{R}^d)\to\mathbb{R}$ is a running cost incurred by an agent based on their interaction with rest of the population, and $G:\mathbb{R}^d\times\mathcal{P}(\mathbb{R}^d)\to\mathbb{R}$ is a terminal cost incurred by an agent based on their final position and the final distribution of the whole population. The terms F and G are called mean field terms because they encode the interaction of a single agent with the rest of the population.

The agents forecast a distribution of the population, $\{\rho(\cdot,t)\}_{t=0}^T$, and aim at minimizing their cost. Therefore, at a Nash equilibrium, we have that, for every $x\in\mathbb{R}^d$,

$$J_{x,0}(v,\rho)\le J_{x,0}(\hat v,\rho), \quad \forall \hat v:[0,T]\to\mathbb{R}^d, \quad [3]$$

where v is the equilibrium strategy of an agent at position x. Here, we assume that agents are small, and their unilateral actions do not alter the density ρ.

From Eq. **3**, we have that individual agents solve an optimal control problem that has a value function

$$\Phi(x,t)=\inf_v J_{x,t}(v,\rho),\ \text{s.t. Eq. 2.} \quad [4]$$

See Fig. 1 for an illustration of an optimal control problem. From the optimal control theory (for details, see ref. 31, sections I.5, I.6, and II.15 or ref. 30, section 10.3), we have that Φ solves the HJB equation

$$-\partial_t\Phi(x,t)+H(x,\nabla\Phi(x,t))=F(x,\rho(x,t)), \\ \Phi(x,T)=G(x,\rho(x,T)), \quad [5]$$

where the Hamiltonian, $H:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$, is defined as

$$H(x,p)=\sup_v -p^\top v-L(x,v). \quad [6]$$

Furthermore, the Pontryagin Maximum Principle yields that the optimal strategy for an agent at position $x\in\mathbb{R}^d$ and time $t\in(0,T]$ is given by the formula

$$v(x,t)=-\nabla_p H(x,\nabla\Phi(x,t)). \quad [7]$$

Assuming that all agents act optimally according to Eq. **7**, the population density, $\hat\rho$, satisfies the continuity equation

$$\partial_t\hat\rho(x,t)-\nabla\cdot(\hat\rho(x,t)\nabla_p H(x,\nabla\Phi(x,t)))=0, \\ \hat\rho(x,0)=\rho_0(x), \quad [8]$$

where $\rho_0\in\mathcal{P}(\mathbb{R}^d)$ is the given population density at time $t=0$. Therefore, an MFG equilibrium is a state when the anticipated distribution coincides with the actual distribution of the population when everyone acts optimally; that is, $\hat\rho=\rho$.

The above discussion suggests an alternative to optimizing the strategy v individually for each agent. One can obtain the optimal strategy for all agents simultaneously by solving the coupled PDE system given by the HJB Eq. **5** and continuity Eq. **8** and then using Eq. **7**. Our work follows this macroscopic approach and aims at reducing the immense computational challenges caused by the high dimension of these PDEs.

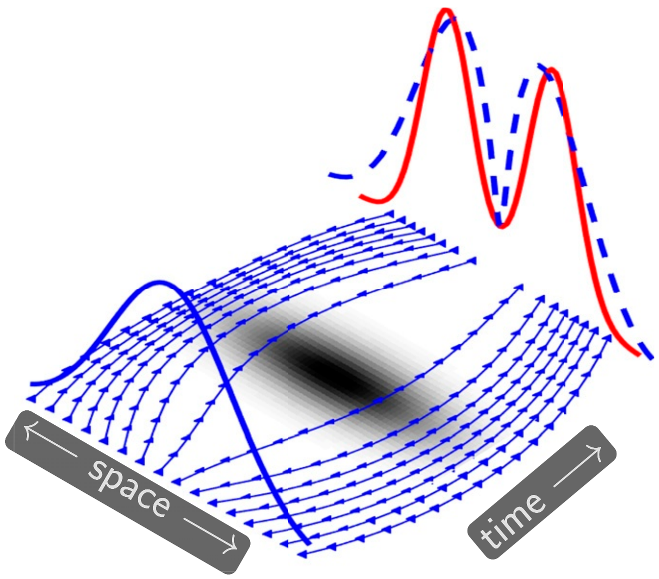

Fig. 1. Illustration of a one-dimensional crowd motion problem. Initially, the crowd of agents is distributed according to ρ_0 (thick blue line) and aims at reaching the target distribution ρ_1 (red solid line) while avoiding the dark regions in the center of the space–time domain. The blue lines marked with arrows depict the agents' trajectories, that is, the characteristics. The dashed blue line depicts the push forward of the initial densities at the final time.

Potential MFGs. Assume that there exist functionals $\mathcal{F},\ \mathcal{G}:\mathcal{P}(\mathbb{R}^d)\to\mathbb{R}$ such that

$$F(x,\rho)=\frac{\delta\mathcal{F}(\rho)}{\delta\rho}(x), \quad G(x,\rho)=\frac{\delta\mathcal{G}(\rho)}{\delta\rho}(x),$$

where $\delta/\delta\rho$ is the variational derivative; that is, for a function $w\in L^2(\mathbb{R}^d)$ and probability measure $\rho(x)dx\in\mathcal{P}(\mathbb{R}^d)$, we have that

$$\lim_{h\to0}\frac{\mathcal{F}(\rho+hw)-\mathcal{F}(\rho)}{h}=\int_{\mathbb{R}^d}F(x,\rho)w(x)dx,$$

and similarly for $\mathcal{G}$.

In the seminal paper (3), Lasry and Lions observed that, in this case, the MFG system Eqs. **5** and **8** coincides with first-order optimality conditions of the infinite-dimensional constrained optimization problem

$$\begin{aligned} \inf_{\rho,v}\quad & \mathcal{J}_{\mathrm{MFG}}(v,\rho)\\ \text{s.t.}\quad & \partial_t\rho(x,t)+\nabla\cdot(\rho(x,t)v(x,t))=0, \quad t\in(0,T]\\ & \rho(x,0)=\rho_0(x), \qquad x\in\mathbb{R}^d \end{aligned} \quad [9]$$

whose objective functional reads

$$\mathcal{J}_{\mathrm{MFG}}(v,\rho)=\int_0^T\int_{\mathbb{R}^d}L(x,v(x,t))\rho(x,t)\,dxdt \\ +\int_0^T\mathcal{F}(\rho(\cdot,t))\,dt+\mathcal{G}(\rho(\cdot,T)). \quad [10]$$

This important class of MFGs is called potential MFGs. Here, the optimal strategies for all agents can be found simultaneously by solving the variational problem Eq. **9**. This is reminiscent of potential games, whose Nash equilibria are critical points of a single function that is called a potential. Remarkably, the Lagrange multiplier associated with the constraint in Eq. **9** satisfies the HJB Eq. **5**.

We use Eq. **7** to reparameterize the control in Eq. **9** and solve the infinite-dimensional optimal control problem

$$\inf_{\rho,\Phi} \mathcal{J}_{\mathrm{MFG}}(-\nabla_p H(x,\nabla\Phi),\rho) + \mathcal{C}_{\mathrm{HJB}}(\Phi,\rho) \text{ s.t. Eq. 8,} \quad \textbf{[11]}$$

where we also added a penalty term for the HJB equation that reads

$$\begin{aligned}\mathcal{C}_{\mathrm{HJB}}(\Phi,\rho) = {}& \alpha_1 \int_0^T \int_{R^d} C_1(\Phi,\rho,x,t)\rho(x,t)\,dxdt \\ & + \alpha_2 \int_{R^d} C_2(\Phi,\rho,x)\rho(x,T)\,dx.\end{aligned} \quad \textbf{[12]}$$

The terms penalize violations of the HJB equation in $(0,T)$ and at the final time and read, respectively,

$$\begin{aligned}& C_1(\Phi,\rho,x,t) \\ & = |\partial_t\Phi(x,t) - H(x,\nabla\Phi(x,t)) + F(x,\rho(x,t))|\end{aligned} \quad \textbf{[13]}$$

$$C_2(\Phi,\rho,x) = |\Phi(x,T) - G(x,\rho(x,T))|. \quad \textbf{[14]}$$

Adding these terms does not affect the minimizer; however, we show, in a numerical experiment, that it improves the convergence of the discretized problem. Since we penalize the violation of the optimality conditions of the original optimization problem, our penalty bears some similarity with Augmented Lagrangian methods. The square of the first term has also been used in ref. 58. Our use of the absolute value in both terms is similar to exact penalty methods (see, e.g., ref. 64, chapter 15). Compared to the equivalent problem Eq. **9**, this formulation enforces Eq. **7** by construction, which can reduce the computational cost of a numerical solution (see ref. 40).

MFC. Mathematically, MFC models are similar to potential MFGs. The key difference in MFC is that a central planner devises an optimal strategy, $v : \mathbb{R}^d \times [0,T] \to \mathbb{R}^d$. All agents in the population follow this strategy in Eq. **2**, resulting in individual cost given by Eq. **1**. The goal of the central player is to minimize the overall costs obtained by replacing the running cost, $\mathcal{F}$, and final cost, $\mathcal{G}$, in Eq. **11** by

$$\int_{\mathbb{R}^d} F(x,\rho(x))\rho(x)\,dx \text{ and } \int_{\mathbb{R}^d} G(x,\rho(x))\rho(x)\,dx, \quad \textbf{[15]}$$

respectively. This choice changes the right-hand sides in the HJB Eq. **5** to

$$\begin{aligned}& F(x,\rho) + \int_{\mathbb{R}^d} \frac{\delta F}{\delta\rho}(\rho)(x)\rho\,dx \\ \text{and} \quad & G(x,\rho) + \int_{\mathbb{R}^d} \frac{\delta G}{\delta\rho}(\rho)(x)\rho\,dx.\end{aligned}$$

These terms are the variational derivatives of the running and final congestion costs in Eq. **15**, respectively. Similar to potential MFGs, finding the optimal strategy v is equivalent to determining the value function Φ by solving Eq. **11**.

Lagrangian Method

We follow a discretize-then-optimize approach, to obtain a finite-dimensional instance of the optimal control problem Eq. **11**. The key idea of our framework is to overcome the curse of dimensionality by using a Lagrangian method to discretize and eliminate the continuity Eq. **8** and all other terms of the objective. We note that the trajectories of the individual agents in Eq. **2** are the characteristic curves of the continuity equation. Hence, Lagrangian coordinates are connected to the microscopic model and are a natural way to describe MFGs. Using Lagrangian coordinates, we obtain a stable and mesh-free discretization that parallelizes trivially.

To keep the discussion concrete and our notation brief, from now on, we consider the L_2 transport costs

$$L(x,v) = \frac{\lambda_{\mathrm{L}}}{2}\|v\|^2, \quad \textbf{[16]}$$

where $\lambda_{\mathrm{L}} > 0$ is a fixed parameter. Using Eq. **6**, it is easy to verify that the Hamiltonian is

$$H(x,p) = \frac{1}{2\lambda_{\mathrm{L}}}\|p\|^2. \quad \textbf{[17]}$$

We emphasize that our framework can be adapted to handle other choices of transport costs.

We solve the continuity Eq. **8** using the method of characteristics and Jacobi's identity (ref. 65, chapter 10). Thereby, we eliminate the density ρ by using the constraint in Eq. **11** and obtain an unconstrained problem whose optimization variable is Φ. Given Φ, we obtain the characteristics by solving the ordinary differential equation (ODE)

$$\begin{aligned}\partial_t z(x,t) &= -\nabla_p H(z(x,t),\nabla\Phi(z(x,t),t)) \\ &= -\frac{1}{\lambda_{\mathrm{L}}}\nabla\Phi(z(x,t),t)\end{aligned} \quad \textbf{[18]}$$

with $z(x,0) = x$. The first step is derived by inserting Eq. **7** into Eq. **2**, and the second step is specific to the H in Eq. **17**. For clarity, we explicitly denote the dependence of the characteristic on the starting point x in the following. Along the curve $z(x,\cdot)$, the solution of the continuity equation satisfies for all $t \in [0,T]$,

$$\rho(z(x,t),t)\det(\nabla z(x,t)) = \rho_0(x). \quad \textbf{[19]}$$

In some cases, for example, OT, the characteristic curves do not intersect, which implies that the mapping $x \mapsto z(x,t)$ is a diffeomorphism and the Jacobian determinant is strictly positive (ref. 66, lemma 7.2.1).

Solving the continuity equation using Eq. **19** requires an efficient way of computing the Jacobian determinant along the characteristics. Direct methods require, in general, $\mathcal{O}(d^3)$ floating point operations (FLOPS), which is intractable when d is large. Alternatively, we follow refs. 57 and 58 and use the Jacobi identity, which characterizes the evolution of the logarithm of the Jacobian determinant along the characteristics, that is,

$$\begin{aligned}\partial_t l(x,t) &= -\nabla\cdot(\nabla_p H(z(x,t),\nabla\Phi(z(x,t),t))) \\ &= -\frac{1}{\lambda_{\mathrm{L}}}\Delta\Phi(z(x,t),t),\end{aligned} \quad \textbf{[20]}$$

with $l(x,0) = 0$. Here, the second step uses Eq. **17**, Δ is the Laplace operator, and we denote $l(x,t) = \log(\det(\nabla z(x,t)))$. This way, we avoid computing the determinant at the cost of numerical integration along the characteristics followed by exponentiation (see also ref. 65). When the determinant is required, we recommend using an accurate numerical integration technique to avoid large errors arising from the exponentiation. However, we shall see below that many problems can be solved using the log-determinant.

An obvious, yet important, observation is that Lagrangian versions of some terms of the optimal control problems do not involve the Jacobian determinant. For example, using Eq. **19** and applying the change of variable formula to the first term in Eq. **10** gives

$$\int_{\mathbb{R}^d} L(x, v(x,t))\rho(x,t)\,dx \qquad [21]$$

$$= \int_{\mathbb{R}^d} L\left(z(x,t), v(z(x,t),t)\right)\rho_0(x)\,dx. \qquad [22]$$

Similarly, the Lagrangian versions of the penalty Eq. **12** do not involve the Jacobian determinant.

Using this expression, we can obtain the accumulated transport costs associated with x as $c_{\rm L}(x,T)\rho_0(x)$, where $c_{\rm L}$ is given by

$$\begin{aligned}\partial_t c_{\rm L}(x,t) &= L(x, -\nabla_p H(x, \nabla\Phi(x,t)))\\ &= \frac{1}{2\lambda_{\rm L}}\|\nabla\Phi(x,t)\|^2.\end{aligned} \qquad [23]$$

Here, $c_{\rm L}(x,0)=0$, and, as before, the second step uses Eq. **17**.

We also accumulate the running costs associated with a fixed x along the characteristics by integrating

$$\partial_t c_{\rm F}(x,t) = \hat{F}(z(x,t), \rho(z(x,t),t), t),$$

where $c_{\rm F}(x,\rho_0(x),0)=0$, and $\hat{F}$ denotes the integrand obtained by applying a change of variables to the functional. As this computation is application-dependent, we postpone it until *Numerical Experiments*.

We note that the trajectories associated with an optimal value function Φ must satisfy the HJB Eq. **5**. One way to ensure this by construction is to integrate the Hamiltonian system as also proposed in ref. 57. Since this complicates the use of the Jacobi identity, we instead penalize violations of the HJB along the characteristics using

$$\partial_t c_1(x,t) = C_1(\Phi, \rho, z(x,t), t), \quad c_1(x,0)=0, \qquad [24]$$

where the right-hand side is given in Eq. **13**. In summary, we compute the trajectories, log-determinant, transportation costs, running costs, and HJB violations by solving the initial value problem

$$\partial_t \begin{pmatrix} z(x,t)\\ l(x,t)\\ c_{\rm L}(x,t)\\ c_{\rm F}(x,t)\\ c_1(x,t)\end{pmatrix} = \begin{pmatrix} -\frac{1}{\lambda_{\rm L}}\nabla\Phi(z(x,t),t)\\ -\frac{1}{\lambda_{\rm L}}\Delta\Phi(z(x,t),t)\\ \frac{1}{2\lambda_{\rm L}}\|\nabla\Phi(z(x,t),t)\|^2\\ \hat{F}(z(x,t),t)\\ C_1(z(x,t),t)\end{pmatrix}. \qquad [25]$$

As discussed above, $z(x,0)=x$ is the origin of the characteristic, and all other terms are initialized with zero. We use the first two components of the ODE to solve the continuity equation and use the last three terms to accumulate the running costs along the characteristics.

Optimization Problem. We now approximate the integrals in Eq. **11** using a quadrature rule. Together with the Lagrangian method Eq. **25**, this leads to an unconstrained optimization problem in the value function Φ, which we will model with a neural network in *ML Framework for MFGs*.

Our approach is modular with respect to the choice of the quadrature; however, as we are mostly interested in high-dimensional cases, we use Monte Carlo integration, that is,

$$\begin{aligned}\min_{\Phi}\ \mathbb{E}_{\rho_0}\Big(&c_{\rm L}(x,T) + c_{\rm F}(x,T) + \hat{G}(z(x,T))\\ &+ \alpha_1 c_1(x,T) + \alpha_2 C_2(\Phi,\rho,z(x,T))\Big),\end{aligned} \qquad [26]$$

where the characteristics are computed using Eq. **25**, and the change of variable used to compute $\hat{G}$ is discussed in *Numerical Experiments*. The above problem consists of minimizing the expected loss function, which is given by the sum of the running costs, terminal costs, and HJB penalty.

Let $x_1,\ldots,x_n \in \mathbb{R}^d$ be random samples from the probability distribution with density $\mu \in \mathcal{P}(\mathbb{R}^d)$; common choices for μ are uniform distribution or $\mu=\rho_0$. Then, summarizing the computations from this section, we obtain the optimization problem

$$\begin{aligned}\min_{\Phi}\sum_{k=1}^{n} v_k\Big(&c_{\rm L}(x_k,T) + c_{\rm F}(x_k,T) + \hat{G}(z(x_k,T))\\ &+ \alpha_1 c_1(x_k,T) + \alpha_2 C_2(\Phi,\rho,z(x_k,T))\Big),\end{aligned} \qquad [27]$$

where the quadrature weight for the measure $I(g) = \int_{\mathbb{R}^d} g\rho_0(x)\,dx$ associated with the kth sample point x_k is

$$v_k = \frac{\rho_0(x_k)}{\mu(x_k)n}. \qquad [28]$$

It is worth reiterating that we transformed the original optimal control problem Eq. **11** in Φ and ρ to an unconstrained optimization problem in Φ. For a given Φ, we eliminate the constraints by solving Eq. **25** independently for each point x_k. In our experiments, we use a fourth-order Runge–Kutta scheme with equidistant time steps. Since the terms of the cost functions can be computed individually, our scheme is trivially parallel.

Optimization algorithms for solving Eq. **27** can roughly be divided into two classes: stochastic approximation (67) and sample average approximation (SAA) (68); see also the recent survey (69). The further class contains, for example, variants of the stochastic gradient scheme. These methods aim at iteratively solving Eq. **26** using a sequence of steps, each of which uses gradient information computed using a relatively small number of sample points. A crucial parameter in these methods is the sequence of step sizes (also known as learning rate) that is typically decaying. When chosen suitably, the steps reduce the expected value in Eq. **26**; however, it is not guaranteed that there exists a step size that reduces the approximation obtained for the current sample points. This prohibits the use of line search or trust region strategies and complicates the application of second-order methods. By contrast, the idea in SAA methods is to use a larger number of points such that Eq. **27** is a suitable approximation of Eq. **26**. Then, the problem can be treated as a deterministic optimization problem and solved, for example, using line search or Trust Region methods. We have experimented with both types of schemes and found an SAA approach based on quasi-Newton schemes with occasional resampling most effective for solving MFG and MFC problems to high accuracy; see *SI Appendix* for an experimental comparison.

ML Framework for MFGs

To obtain a finite-dimensional version of Eq. **27**, we parameterize the value function using a neural network. This enables us to penalize violations of the HJB Eq. **5** during training and thereby ensure that the trained neural network accurately captures the optimality conditions. Using a neural network in the Lagrangian method gives us a mesh-free scheme. In the following, we give a detailed description of our neural network models and the computation of the characteristics.

Neural Network Models for the Value Function. We now introduce our neural network parameterization of the value function. While this is not the only possible parameterization, using neural networks to solve high-dimensional PDE problems has become increasingly common (see, e.g., refs. 49–52). The novelty of our approach is the design of the neural network architecture such that the characteristics in Eq. **25** can be approximated accurately and efficiently.

Our network models map the input vector $s = (x, t) \in \mathbb{R}^{d+1}$ to the value function $\Phi(s)$. In the following, we denote our model as $\Phi(s, \theta)$, where θ is a vector of network parameters (also called weights) to be defined below.

The neural network processes the input features using a number of layers, each of which combines basic operations such as affine transformations and element-wise nonlinearities. While the size of the input and output features is determined by our application, there is flexibility in choosing the size of the hidden layers, which is also called their widths. Altogether, the number, widths, and specific design of the layers are referred to as the network's architecture.

Our base architecture is defined as follows:

$$\begin{aligned}\Phi(s,\theta) = w^\top N(s,\theta_N) + \frac{1}{2} s^\top \left(A + A^\top\right) s + c^\top s + b, \\ \theta = (w, \theta_N, \mathrm{vec}(A), c, b),\end{aligned} \qquad [29]$$

where, for brevity, θ collects the trainable weights $w \in \mathbb{R}^m$, $\theta_N \in \mathbb{R}^p$, $A \in \mathbb{R}^{(d+1)\times(d+1)}$, and $c \in \mathbb{R}^{d+1}$, $b \in \mathbb{R}$. The function N can be any neural network with $(d+1)$ input features, m output features, and parameters $\theta_N \in \mathbb{R}^p$. The last two terms allow us to express quadratic value functions easily. Our approach is modular with respect to the network architecture. However, the design of the neural network's architecture is known to affect its expressibility and the ease of training (see, e.g., ref. 70). It is important to note that the use of the neural network renders Eq. **27**, in general, nonconvex.

In this work, our network architecture is a residual network (ResNet) (71). For a network with M layers and a given input feature $s \in \mathbb{R}^{d+1}$, we obtain $N(s, \theta_N) = u_M$ as the final step of the forward propagation

$$\begin{aligned} u_0 &= \sigma(K_0 s + b_0) \\ u_1 &= u_0 + h\sigma(K_1 u_0 + b_1) \\ \vdots \quad & \qquad \vdots \\ u_M &= u_{M-1} + h\sigma(K_M u_{M-1} + b_M), \end{aligned} \qquad [30]$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is an element-wise activation function, $h > 0$ is a fixed step size, and the network's weights are $K_0 \in \mathbb{R}^{m\times(d+1)}$, $K_1, \ldots, K_M \in \mathbb{R}^{m\times m}$, and $b_0, \ldots, b_M \in \mathbb{R}^m$. In our experiments, we use the activation function

$$\sigma(x) = \log(\exp(x) + \exp(-x)), \qquad [31]$$

which can be seen as a smoothed out absolute value. For notational convenience, we vectorize and concatenate all weights in the vector $\theta_N \in \mathbb{R}^p$. In theory, the larger the number of layers and the larger their width, the more expressive the resulting network. In practice, the full expressiveness may not be realized when the learning problem becomes too difficult and numerical schemes find only suboptimal weights. In our numerical experiments, we found a relatively shallow network with as little as $M = 1$ layers and widths of 16 to be very effective; however, this architecture may not be optimal for other learning problems. The main difference between the forward propagation through a ResNet and a more traditional multilayer perceptron is the addition of the previous feature vector in the computation of $u_1, \ldots, u_M$.

ResNets have been tremendously successful in a wide range of ML tasks and have been found to be trainable even for large depths (i.e., $M \gg 0$). By interpreting Eq. **30** as a forward Euler discretization of an initial value problem, the continuous limit of a ResNet is amenable to mathematical analysis (72, 73). This observation has received a lot of attention recently and been used, for example, to derive maximum principle (74), propose stable ResNet variants (73), and accelerate the training using multilevel (75) and parallel-in-time schemes (76).

Characteristics Computations. To compute the characteristic and other quantities in Eq. **25**, we need to compute the gradient and Laplacian of Φ with respect to the input features. Perhaps the simplest option is to use automatic differentiation (AD), which has become a ubiquitous and mature technology in most ML packages. We note that this convenience comes at the cost of d separate evaluations of directional derivatives when computing the Laplacian. While trace estimation techniques can lower the number of evaluations, this introduces inaccuracies in the PDE solver. Also, we show below that computing the Laplacian exactly is feasible for our network.

We now provide a detailed derivation of our gradient and Laplacian computation. The gradient of our model in Eq. **29** with respect to the input feature s is given by

$$\nabla_s \Phi(s,\theta) = \nabla_s N(s,\theta_N) w + (A + A^\top)s + c. \qquad [32]$$

Due to the ordering in $s = (x, t)$, the first d component of this gradient corresponds to the spatial derivatives of the value function, and the final one is the time derivative. We compute the gradient of the neural network Eq. **30** in the direction w using back-propagation (also called reverse mode differentiation),

$$\begin{aligned} z_M &= w + hK_M^\top \mathsf{diag}\,(\sigma'(K_M u_{M-1} + b_M)) w, \\ \vdots \quad & \qquad \vdots \\ z_1 &= z_2 + hK_1^\top \mathsf{diag}\,(\sigma'(K_1 u_0 + b_1)) z_2, \\ z_0 &= K_0^\top \mathsf{diag}\,(\sigma'(K_0 s + b_0)) z_1, \end{aligned} \qquad [33]$$

which gives $\nabla_s N(s,\theta_N) w = z_0$. Here, $\mathsf{diag}\,(v) \in \mathbb{R}^{m\times m}$ is a diagonal matrix with diagonal elements given by $v \in \mathbb{R}^m$, and $\sigma'(\cdot)$ is computed element-wise.

Next, we compute the Laplacian of the value function model with respect to x. We first note that

$$\Delta\Phi(s,\theta) = \mathrm{tr}\left(E^\top(\nabla_s^2(N(s,\theta_N)w) + (A + A^\top))E\right),$$

where the columns of $E \in \mathbb{R}^{(d+1)\times d}$ are given by the first d standard basis vectors in $\mathbb{R}^{d+1}$. Computing the terms involving A is trivial, and we now discuss how to compute the Laplacian of the neural network in one forward pass through the layers. We first note that the trace of the first layer in Eq. **30** is

$$\begin{aligned} t_0 &= \mathrm{tr}\left(E^\top \nabla_s (K_0^\top \mathsf{diag}\,(\sigma'(K_0 s + b_0)) z_1) E\right) \\ &= \mathrm{tr}\left(E^\top K_0^\top \mathsf{diag}\,(\sigma''(K_0 s + b_0) \odot z_1) K_0 E\right) \\ &= (\sigma''(K_0 s + b_0) \odot z_1)^\top ((K_0 E) \odot (K_0 E))\mathbf{1}, \end{aligned} \qquad [34]$$

where $\odot$ denotes a Hadamard product (i.e., an element-wise product of equally sized vectors or matrices), $\mathbf{1} \in \mathbb{R}^d$ is a vector of all ones, and, in the last, we used that the middle term is a diagonal matrix. The above computation shows that the Laplacian of the first layer can be computed using $\mathcal{O}(m \cdot d)$ FLOPS by first squaring the elements in the first d columns of K_0, then summing those columns, and, finally, one inner product. The computational costs are essentially equal to performing one single matrix–vector product with the Hessian using AD but produces the exact Laplacian.

To compute the Laplacian of the entire ResNet, we continue with the remaining rows in Eq. **33** in reverse order to obtain

$$\Delta(N(s,\theta_N)w) = t_0 + h\sum_{i=1}^{M} t_i, \qquad \textbf{[35]}$$

where t_i is computed as

$$\begin{aligned} t_i &= \mathrm{tr}\left(J_{i-1}^\top \nabla_s (K_i^\top \mathsf{diag}\,(\sigma'(K_i u_{i-1}(s)+b_i)) z_{i+1}) J_{i-1}\right) \\ &= \mathrm{tr}\left(J_{i-1}^\top K_i^\top \mathsf{diag}\,(\sigma''(K_i u_{i-1}+b_i)\odot z_{i+1}) K_i J_{i-1}\right) \\ &= (\sigma''(K_i u_{i-1}+b_i)\odot z_{i+1})^\top ((K_i J_{i-1})\odot(K_i J_{i-1}))\mathbf{1}. \end{aligned}$$

Here, $J_{i-1} = \nabla_s u_{i-1}^\top \in \mathbb{R}^{m\times d}$ is a Jacobian matrix, which can be updated and overwritten in the forward pass at a computational cost of $\mathcal{O}(m^2\cdot d)$ FLOPS.

To summarize the above derivations, we note that each time step of the numerical approximation of the characteristics involves one forward propagation (Eq. **30**), one backpropagation (Eq. **33**), and the trace computations (Eq. **35**). The overall computational costs scale as $\mathcal{O}(m^2\cdot d\cdot M)$, that is, linearly with respect to the input dimension and number of layers, but quadratically with the width of the network. This motivates the use of deep and not wide architectures.

Numerical Experiments

We apply our method to two prototype MFG instances. Here, we give a concise overview of the problems and results and refer to *SI Appendix* for a more detailed description of the problem instances and results. We perform our experiments using a prototype of our algorithm that we implemented in the Julia programming language (48) as an extension of the ML framework Flux (77). Readers can access all codes, scripts, and produced results at http://github.com/EmoryMLIP/MFGnet.jl.

Example 1: Dynamical OT. Given two densities $\rho_0, \rho_1 \in \mathcal{P}(\mathbb{R}^d)$, the OT problem consists of finding the transformation $y:\mathbb{R}^d \to \mathbb{R}^d$ with the smallest transport costs such that the push forward of ρ_0 equals ρ_1. The problem was introduced by Monge (45), revolutionized by the contributions by Kantorovich (78) and Benamou and Brenier (45). Other notable theoretical advances include refs. 46 and 79–81.

Among the many versions of OT, we consider the fluid dynamics formulation (45), which can be seen as a potential MFG. This formulation is equivalent to a convex optimization problem, which can be solved accurately and efficiently if $d \le 3$ (see, e.g., ref. 54). The curse of dimensionality limits applications of these methods when $d > 3$. Approximately solving high-dimensional OT problems is of key interest to ML and Bayesian statistics (see some related works in ref. 58).

We model the fluid dynamic version of OT in ref. 45 as a potential MFG by using

$$\mathcal{F}(\rho) = 0, \quad \mathcal{G}(\rho) = \lambda_{\mathrm{KL}}\mathcal{G}_{\mathrm{KL}}(\rho),$$

where $\lambda_{\mathrm{KL}} > 0$ is a penalty parameter, and the second term is the Kullback–Leibler divergence, which penalizes violations of the final time constraint $\rho(\cdot,T) = \rho_1(\cdot)$ and reads

$$\begin{aligned} \mathcal{G}_{\mathrm{KL}}(\rho) &= \int_{\mathbb{R}^d} \rho(x,T)\log\frac{\rho(x,T)}{\rho_1(x)}dx \\ &= \int_{\mathbb{R}^d} \log\frac{\rho(z(x,T),T)}{\rho_1(z(x,T))}\rho_0(x)dx \\ &= \int_{\mathbb{R}^d} (\log\rho_0(x) - l(x,T) - \log\rho_1(z(x,T)))\rho_0(x)dx. \end{aligned} \qquad \textbf{[36]}$$

Here, the log-determinant, l, is given in Eq. **20**. The variational derivative of this loss function, which is used in the computation of the penalty term in Eq. **14**, is

$$G_{\mathrm{KL}}(x,\rho) = 1 + \log(\rho(x)) - \log\rho_1(x).$$

Note that the optimal control problem Eq. **11** is symmetric with respect to the roles of ρ_0 and ρ_1 if and only if $\lambda_{\mathrm{KL}} = \infty$, because we relaxed the final time constraint.

We generate a synthetic test problem that enables us to increase the dimension of the problem with only minimal effect on its solution. For a given dimension d, we choose the density of the Gaussian white noise distribution as the target

$$\rho_1(x) = \rho_G\,(x, \mathbf{0}, 0.3\cdot\mathbf{I}).$$

Here, $\rho_G(\cdot,\mathbf{m},\mathbf{\Sigma})$ is the probability density function of a d-variate Gaussian with mean $\mathbf{m}\in\mathbb{R}^d$ and covariance matrix $\mathbf{\Sigma}\in\mathbb{R}^{d\times d}$. The initial density is the Gaussian mixture

$$\rho_0(x) = \frac{1}{8}\sum_{j=1}^{8}\rho_G\,(x,\mathbf{m}_j, 0.3\cdot\mathbf{I}),$$

where the means of the individual terms are equally spaced on the intersection of the sphere with radius four and the coordinate plane in the first two space dimensions, that is,

$$\mathbf{m}_j = 4\cdot\cos\left(\frac{2\pi}{8}j\right)\mathbf{e}_1 + 4\cdot\sin\left(\frac{2\pi}{8}j\right)\mathbf{e}_2, \quad j=1,\dots,8.$$

Here, $\mathbf{e}_1$ and $\mathbf{e}_2$ are the first two standard basis vectors. The 2D instances of these densities are visualized in Fig. 2.

We perform four experiments to show the scalability to high-dimensional instances, explore the benefits of the penalty function $\mathcal{C}_{\mathrm{HJB}}$, compare two optimization strategies, and validate our scheme by comparing it to a Eulerian method in $d=2$, respectively. The network architecture is almost exactly the same in all cases; see *SI Appendix* for details.

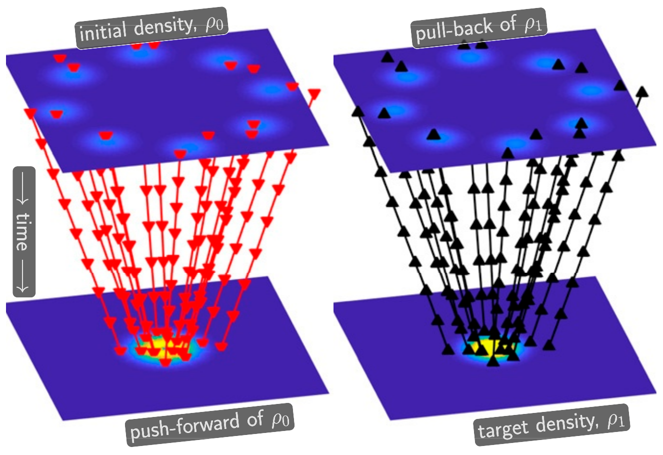

Fig. 2. Visualization of a 2D OT experiment. (*Left*) The given initial density ρ_0 (*Top*) and its push forward at time $t=1$ (*Bottom*). The red lines represent the characteristics starting from randomly sampled points according to ρ_0. (*Right*) The given target density, ρ_1 (*Bottom*) and its pullback (*Top*). The black line corresponds to the characteristics computed backward in time from the endpoints of the red characteristics. The similarity of the images in each row and the fact that the characteristics are almost straight lines show the success of training. Also, since the red and black characteristics are nearly identical, the transformation is invertible. The same color axis is used for all plots.

We show, in Fig. 2, that our network provides a meaningful solution for the 2D instance. The performance can be seen by the close match of the given ρ_0 and the pullback of ρ_1, which is optimized as it enters the terminal costs, and the similarity of the push forward of ρ_0 and ρ_1, which is computed after training. Also, the characteristics are approximately straight. To improve the visualization, we compute the characteristics using 4 times as many time integration steps as used in training. A close inspection of the push-forward density also shows that the network learns to partition the target density into eight approximately equally sized slices, as predicted by the theory (*SI Appendix*, Fig. S4).

Our method obtains quantitatively similar results in higher-dimensional instances ($d = 10, 50, 100$), as can be seen in the summary of the objective function values provided in the top half of Table 1 and the convergence plots in *SI Appendix*, Fig. S5. The values are computed using the validation sets. The table also shows that, despite a moderate growth of the number of training samples, the actual runtime per iteration of our prototype implementation per iteration grows slower than expected from our complexity analysis. We use more samples in higher dimensions to avoid overfitting. Due to the design of the problem, similar transport costs and terminal costs are to be expected. There is a slight increase of the terminal costs for the 50-dimensional instances, but we note that, at least for projections onto the first two coordinate dimensions, the image quality is similar for all cases (*SI Appendix*, Fig. S4).

In our second experiment, we show that, without the use of the HJB penalty, $\mathcal{C}_{\mathrm{HJB}}$, the optimization can fail to match the densities and result in curved characteristics, which are not meaningful in OT (*SI Appendix*, Fig. S6). Increasing the number of time steps to discretize the characteristics, Eq. **25**, improves the results; however, the results are still inferior to the ones obtained with the penalty, and the computational cost of training is 4 times higher. Hence, in this example, using the penalty function, we obtain a more accurate solution at reduced computational costs.

Our third OT experiment compares two optimization strategies: the SAA approach with the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method used throughout our experiments and an SA approach with the Adam method that is common in ML. We note that, for the 2D instance of our problem, Adam converges considerably slower and is less effective in reducing the HJB penalty at a comparable computational cost; however, both methods lead to similar final results.

In our fourth experiment, we compare our proposed method to a provably convergent Eulerian solver for the 2D instance. We compare the optimal controls obtained using both methods using an explicit finite volume solver for the continuity equation, which was not used during the optimization. This step is essential to obtain a fair comparison. *SI Appendix*, Figs. S10 and S12 and Table S2 show that, for this example, our method is competitive and, as demonstrated above, scales to dimensions that are beyond reach with Eulerian schemes.

Table 1. Overview of numerical results for instances of the OT and crowd motion problem in growing space dimensions

Example						
Example 1: OT						
2	2,304	9.99e+00	—	7.01e-01	1.17e+00	2.038
10	6,400	1.01e+01	—	8.08e-01	1.21e+00	8.256
50	16,384	1.01e+01	—	6.98e-01	2.94e+00	81.764
100	36,864	1.01e+01	—	8.08e-01	1.21e+00	301.043
Example 2: Crowd Motion						
2	2,304	1.65e+01	2.29e+00	7.81e-01	2.27e-01	4.122
10	6,400	1.65e+01	2.22e+00	7.51e-01	3.94e-01	17.205
50	9,216	1.65e+01	1.91e+00	7.20e-01	1.21e+00	134.938
100	12,544	1.65e+01	1.49e+00	1.00e+00	2.78e+00	241.727

All values were approximated using the validation points; *d*, space dimension; *n*, number of training samples; $\mathcal{L}$, transport costs; $\mathcal{F}$, running costs; $\mathcal{G}$, terminal costs; $\mathcal{C}_{\mathrm{HJB}}$, HJB penalty.

Example 2: Crowd Motion. We consider the motion of a crowd of agents distributed according to an initial density ρ_0 to the desired state ρ_1. In contrast to the OT problem, the agents trade off reaching ρ_1 with additional terms that encode their spatially varying preference and their desire to avoid crowded regions.

To force the agents to move to the target distribution, we use the same terminal cost as in the previous example. To express the agents' preference to avoid congestion and model costs associated with traveling through the center of the domain, we use the mean field potential energy

$$\mathcal{F}(\rho(\cdot,t)) = \lambda_{\mathrm{E}}\mathcal{F}_{\mathrm{E}}(\rho(\cdot,t)) + \lambda_{\mathrm{P}}\mathcal{F}_{\mathrm{P}}(\rho(\cdot,t)), \qquad \textbf{[37]}$$

which is a weighted sum of an entropy and preference term defined, respectively, as

$$\mathcal{F}_{\mathrm{E}}(\rho(\cdot,t)) = \int_{\mathbb{R}^d} \rho(x,t)\log\rho(x,t)\,dx,$$
$$\mathcal{F}_{\mathrm{P}}(\rho(\cdot,t)) = \int_{\mathbb{R}^d} Q(x)\rho(x,t)\,dx.$$

The entropy terms penalizes the accumulation of agents; that is, the more spread out the agents are, the smaller this term. In the third term, $Q:\mathbb{R}^d \to \mathbb{R}$, models the spatial preferences of agents; that is, the smaller $Q(x)$, the more desired is the position x. Carrying out similar steps to compute these terms in Lagrangian coordinates yields

$$\mathcal{F}_{\mathrm{E}}(\rho(\cdot,t)) = \int_{\mathbb{R}^d} (\log\rho_0(x) - l(x,t))\rho_0(x)\,dx, \qquad \textbf{[38]}$$
$$\mathcal{F}_{\mathrm{P}}(\rho(\cdot,t)) = \int_{\mathbb{R}^d} Q(z(x,t))\rho_0(x)\,dx. \qquad \textbf{[39]}$$

In our experiment, we control the relative influence of both terms by choosing $\lambda_{\mathrm{E}} = 0.01, \lambda_{\mathrm{P}} = 1$, respectively, in Eq. **37**. To penalize the L_2 transport costs, we use the Lagrangian and Hamiltonian given in Eqs. **16** and **17**.

Finally, we take the variational derivative to obtain the HJB Eq. **5**. We note that the mean field coupling is

$$\frac{\delta\mathcal{F}(\rho)}{\delta\rho} = F(x,\rho) = \lambda_{\mathrm{F}}F_{\mathrm{E}}(x,\rho) + \lambda_{\mathrm{P}}F_{\mathrm{P}}(x,\rho),$$

with the terms

$$F_{\mathrm{E}}(x,\rho) = \log\rho(x) + 1,$$
$$F_{\mathrm{P}}(x,\rho) = Q(x).$$

A detailed description of the experimental setup is provided in *SI Appendix*. The initial and target densities are shifted Gaussians

$$\rho_0 = \rho_G(x, 3\cdot\mathbf{e}_2, 0.3\cdot\mathbf{I}), \quad \rho_1(x) = \rho_G(x, -3\cdot\mathbf{e}_2, 0.3\cdot\mathbf{I}).$$

Note that, for $\lambda_{\mathrm{E}} = \lambda_{\mathrm{P}} = 0$, the problem becomes a trivial OT problem, and the optimal trajectories would be straight and parallel through the center of the domain. To obtain a more interesting dynamic, we introduce the preference function

$$Q(x) = 50\cdot\rho_G\left(x, \mathbf{0}, \mathsf{diag}\left(1, 0.5\right)\right).$$

Since Q attains its maximum at the origin, agents have an incentive to avoid this region, which leads to curved characteristics. As terminal costs, we use the Kullback–Leibler divergence Eq. **36**.

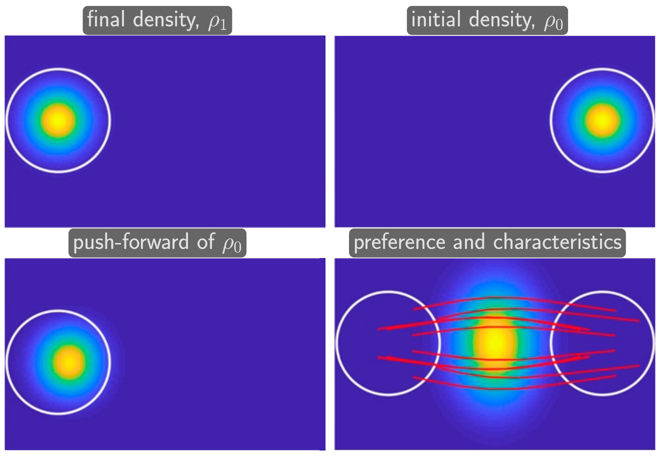

Fig. 3. Illustration of the 2D crowd motion problem. The initial density (*Top Left*) and target density (*Bottom Left*) are Gaussians centered at the top or bottom, respectively, of the domain. The white circles depict a contour line for the density and are added in the remaining subplots to help visual assessment. The agents' goal is to move near the target while avoiding congestion and the obstacle in the center of the domain (*Top Right*). There are higher costs associated with traveling through regions where the preference function is large (represented by yellow regions). The red lines in *Top Right* show the learned trajectories. It can be seen that the characteristics are curved to avoid the center of the domain. We also show that the push-forward of the initial density is similar to the target density (*Bottom Right*).

For the higher-dimensional instances, we evaluate the preference function using the first two components of x. Thereby, the preference function becomes invariant to the remaining entries, and the solutions become comparable across dimensions.

As in the OT example, we obtain similar but not identical objective function values for the problem instances obtained by choosing $d \in \{2, 10, 50, 100\}$; see the bottom half of Table 1. Again, we increased the number of training samples with dimension, and we observe that the runtime of our prototype code scales better than predicted. Fig. 3 shows the optimized trajectories for the $d = 2$ instance; see *SI Appendix*, Fig. S9 for similar visualizations for the remaining instances. As expected, the agents avoid the crowded regions and prefer to spread out horizontally to avoid congestion. We chose the starting points of the characteristics to be symmetric about the x_2 axis. As expected, this renders the learned trajectories approximately symmetric as well. We note that the characteristics are visually similar across dimensions and that our results are comparable to those obtained with a provably convergent Eulerian solver for the 2D instance (*SI Appendix*, Figs. S11 and S13 and Table S2).

Discussion and Outlook

By combining Lagrangian PDE solvers with neural networks, we develop a framework for the numerical solution of potential MFGs and MFC problems. Our method is geared toward high-dimensional problem instances that are beyond reach with existing solution methods. Since our method is mesh-free and well suited for parallel computation, we believe it provides a promising direction toward much-anticipated large-scale applications of MFGs. Even though our scheme is competitive with Eulerian schemes based on convex optimization for the 2D examples, its main advantage is the scalability to higher dimensions. We exemplify the effectiveness of our method using an OT and a crowd motion problem in 100 dimensions. Using the latter example, we also demonstrate that our scheme can learn complex dynamics even with a relatively simple neural network model.

The fact that our framework transforms MFGs into types of ML problems brings exciting opportunities to advance MFG application and theory. While we can already leverage the many advances made in ML over the last decades, the learning problem in MFGs has some unique traits that require further attention.

Open mathematical issues include the development of practical convergence results for neural networks in optimal control. In fact, it is not always clear that a larger network will perform better in practice. Our numerical experiment for the OT problem makes us optimistic that our framework may be able to solve HJB equations in high dimensions to practically relevant accuracy. Similarly, the impact of regularization parameters on the optimization deserves further attention. However, more analysis is needed to obtain firm theoretical results for this property.

A critical computational issue is a more thorough parallel implementation, which may provide the speed-up needed to solve more-realistic MFG problems. Also, advances in numerical optimization for deep learning problems may help solve the learning problems more efficiently.

Open questions on the ML side include the setup of the learning problem. There are little to no theoretical guidelines that help design network architectures that generalize well. Although there is a rich body of examples for data science applications, our learning problem has different requirements. For example, not only does the neural network need to be evaluated, but the Lagrangian scheme also requires its gradient and Laplacian. A careful choice of the architecture may be necessary to ensure the required differentiability of the network. Also, more experiments are needed to establish deeper intuition into the interplay between the network architecture and other hyperparameters, for example, the choice of the penalty parameter or the time discretization of the characteristics.

An obvious open question from an application perspective is the use of our framework to solve more-realistic problems. To promote progress in this area, we provide our code under a permissible open source license.

ACKNOWLEDGMENTS. We thank Derek Onken for fruitful discussions and his help with the Julia implementation; and Wonjun Lee and Siting Liu for their assistance with the implementation of the Eulerian solver for the 2D problem instances and their suggestions that helped improve the manuscript. L.R. is supported by NSF Grant DMS-1751636. This research was performed while L.R. was visiting the Institute for Pure and Applied Mathematics, Los Angeles, CA, which is supported by NSF Grant DMS-1440415. S.J.O., W.L., L.N., and S.W.F. receive support from Air Force Office of Science Research Grants MURI-FA9550-18-1-0502 and FA9550-18-1-0167, and Office of Naval Research Grant N00014-18-1-2527.

1. J.-M. Lasry, P.-L. Lions, Jeux à champ moyen. I. Le cas stationnaire. *C. R. Math. Acad. Sci. Paris* **343**, 619–625 (2006).
2. J.-M. Lasry, P.-L. Lions, Jeux à champ moyen. II. Horizon fini et contrôle optimal. *C. R. Math. Acad. Sci. Paris* **343**, 679–684 (2006).
3. J.-M. Lasry, P.-L. Lions, Mean field games. *Jpn. J. Math.* **2**, 229–260 (2007).
4. M. Huang, R. P. Malhamé, P. E. Caines, Large population stochastic dynamic games: Closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Commun. Inf. Syst.* **6**, 221–251 (2006).
5. M. Huang, P. E. Caines, R. P. Malhamé, Large-population cost-coupled LQG problems with nonuniform agents: Individual-mass behavior and decentralized ϵ-Nash equilibria. *IEEE Trans. Autom. Control* **52**, 1560–1571 (2007).
6. A. Bensoussan, J. Frehse, P. Yam, *Mean Field Games and Mean Field Type Control Theory* (Springer, New York, 2013).
7. O. Guéant, J.-M. Lasry, P.-L. Lions, "Mean field games and applications" in *Paris-Princeton Lectures on Mathematical Finance 2010*, R. Carmona *et al.*, Eds. (Lecture Notes in Mathematics, Springer, Berlin, 2011), vol. 2003, pp. 205–266.
8. Y. Achdou, F. J. Buera, J.-M. Lasry, P.-L. Lions, B. Moll, Partial differential equation models in macroeconomics. *Philos. Trans. R. Soc. A* **372**, 20130397 (2014).
9. D. A. Gomes, L. Nurbekyan, E. A. Pimentel, *Economic Models and Mean-Field Games Theory* (Instituto Nacional de Matemática Pura e Aplicada, Rio de Janeiro, Brazil, 2015).

10. Y. Achdou, J. Han, J.-M. Lasry, P.-L. Lions, B. Moll, Income and wealth distribution in macroeconomics: A continuous-time approach (Working pap. 23732, National Bureau of Economic Research, 2017), 10.3386/w23732.
11. A. Lachapelle, J.-M. Lasry, C.-A. Lehalle, P.-L. Lions, Efficiency of the price formation process in presence of high frequency participants: A mean field game analysis. *Math. Financ. Econ.* **10**, 223–262 (2016).
12. P. Cardaliaguet, C.-A. Lehalle, Mean field game of controls and an application to trade crowding. *Math. Financ. Econ.* **12**, 335–363 (2018).
13. P. Casgrain, S. Jaimungal, Algorithmic trading in competitive markets with mean field games. *SIAM News* **52**, 1–2 (2019).
14. D. Firoozi, P. E. Caines, "An optimal execution problem in finance targeting the market trading speed: An MFG formulation" in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* (Institute of Electrical and Electronics Engineers, 2017), pp. 7–14.
15. A. Lachapelle, M.-T. Wolfram, On a mean field game approach modeling congestion and aversion in pedestrian crowds. *Transp. Res. Part B Methodol.* **45**, 1572–1589 (2011).
16. M. Burger, M. Di Francesco, P. A. Markowich, M.-T. Wolfram, Mean field games with nonlinear mobilities in pedestrian dynamics. *Discrete Contin. Dyn. Syst. Ser. B* **19**, 1311–1333 (2014).
17. A. Aurell, B. Djehiche, Mean-field type modeling of nonlocal crowd aversion in pedestrian crowd dynamics. *SIAM J. Contr. Optim.* **56**, 434–455 (2018).
18. Y. Achdou, J.-M. Lasry, "Mean field games for modeling crowd motion" in *Contributions to Partial Differential Equations and Applications*, B. Chetverushkin *et al.*, Eds. (Computational Methods in Applied Sciences, Springer, Cham, Switzerland, 2019), vol. 47, pp. 17–42.
19. A. C. Kizilkale, R. Salhab, R. P. Malhamé, An integral control formulation of mean field game based large scale coordination of loads in smart grids. *Automatica* **100**, 312–322 (2019).
20. A. De Paola, V. Trovato, D. Angeli, G. Strbac, A mean field game approach for distributed control of thermostatic loads acting in simultaneous energy-frequency response markets. *IEEE Transactions Smart Grid* **10**, 5987–5999 (2019).
21. D. A. Gomes, J. Saúde, A mean-field game approach to price formation in electricity markets. arXiv:1807.07088 (19 November 2018).
22. W. E, J. Han, Q. Li, A mean-field optimal control formulation of deep learning, arXiv:1807.01083 (3 July 2018).
23. P. M. Welch, K. Ø. Rasmussen, C. F. Welch, Describing nonequilibrium soft matter with mean field game theory. *J. Chem. Phys.* **150**, 174905 (2019).
24. P. Cardaliaguet, F. Delarue, J.-M. Lasry, P.-L. Lions, *The Master Equation and the Convergence Problem in Mean Field Games* (Annals of Mathematics, Princeton University Press, Princeton, NJ, 2019), vol. 201.
25. W. Gangbo, A. Święch, Existence of a solution to an equation arising from the theory of mean field games. *J. Differ. Equ.* **259**, 6573–6643 (2015).
26. D. A. Gomes, E. A. Pimentel, V. Voskanyan, *Regularity Theory for Mean-Field Game Systems Regularity Theory for Mean-Field Game Systems* (Springer, Cham, Switzerland, 2016).
27. A. Cesaroni, M. Cirant, "Introduction to variational methods for viscous ergodic mean-field games with local coupling" in *Contemporary Research in Elliptic PDEs and Related Topics*, S. Dipierro, Ed. (INdAM Series, Springer, Cham, Switzerland, 2019), vol. 33, pp. 221–246.
28. R. Carmona, F. Delarue, *Probabilistic Theory of Mean Field Games with Applications. I* (Probability Theory and Stochastic Modelling, Springer, Cham, Switzerland, 2018), vol. 83.
29. R. Carmona, F. Delarue, *Probabilistic Theory of Mean Field Games with Applications. II* (Probability Theory and Stochastic Modelling, Springer, Cham, Switzerland, 2018), vol. 84.
30. L. C. Evans, *Partial Differential Equations* (Graduate Studies in Mathematics, American Mathematical Society, Providence, RI, ed. 2, 2010), vol. 19.
31. W. H. Fleming, H. M. Soner, *Controlled Markov Processes and Viscosity Solutions* (Stochastic Modelling and Applied Probability, Springer, New York, NY, ed. 2, 2006), vol. 25.
32. Y. Achdou, "Finite difference methods for mean field games" in *Hamilton-Jacobi Equations: Approximations, Numerical Analysis and Applications*, P. Loreti, N. Tchou, Eds. (Lecture Notes in Mathematics, Springer, Heidelberg, Germany, 2013), vol. 74, pp. 1–47.
33. E. Carlini, F. J. Silva, A semi-Lagrangian scheme for a degenerate second order mean field game system. *Discrete Contin. Dyn. Syst.* **35**, 4269–4292 (2015).
34. Y. Achdou, M. Laurière, Mean field type control with congestion (II): An augmented Lagrangian method. *Appl. Math. Optim.* **74**, 535–578 (2016).
35. N. Almulla, R. Ferreira, D. Gomes, Two numerical approaches to stationary mean-field games. *Dyn. Games Appl.* **7**, 657–682 (2017).
36. J.-D. Benamou, G. Carlier, F. Santambrogio, "Variational mean field games" in *Active Particles*, N. Bellomo, P. Degond, E. Tadmor, Eds. (Advances in Theory, Models, and Applications, Modeling and Simulation in Science, Engineering and Technology, Birkhäuser/Springer, Cham, Switzerland, 2017), vol. 1, pp. 141–171.
37. J.-D. Benamou, G. Carlier, S. Di Marino, L. Nenna, An entropy minimization approach to second-order variational mean-field games. *Math. Model Methods Appl. Sci.* **29**, 1553–1583 (2019).
38. D. Evangelista, R. Ferreira, D. A. Gomes, L. Nurbekyan, V. Voskanyan, First-order, stationary mean-field games with congestion. *Nonlinear Anal.* **173**, 37–74 (2018).
39. M. Jacobs, F. Léger, A fast approach to optimal transport: The back-and-forth method. arXiv:1905.12154 (29 May 2019).
40. Y. T. Chow, W. Li, S. Osher, W. Yin, Algorithm for Hamilton–Jacobi equations in density space via a generalized Hopf formula. *J. Sci. Comput.* **80**, 1195–1239 (2019).
41. L. Briceño Arias *et al.*, "On the implementation of a primal-dual algorithm for second order time-dependent mean field games with local couplings" in *CEMRACS 2017—Numerical Methods for Stochastic Models: Control, Uncertainty Quantification, Mean-Field*, B. Bouchard, J.-F. Chassagneux, F. Delarue, E. Gobet, J. Lelong, Eds. (ESAIM Proceedings and Surveys, EDP Science, Les Ulis, France, 2019), vol. 65, pp. 330–348.
42. M. Jacobs, F. Léger, W. Li, S. Osher, Solving large-scale optimization problems with a convergence rate independent of grid size. *SIAM J. Numer. Anal.* **57**, 1100–1123 (2019).
43. R. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, NJ, 1957).
44. L. Nurbekyan, J. Saúde, Fourier approximation methods for first-order nonlocal mean-field games. *Port. Math.* **75**, 367–396 (2018).
45. J. D. Benamou, Y. Brenier, A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numer. Math.* **84**, 375–393 (2000).
46. C. Villani, *Optimal Transport: Old and New* (Springer Science, 2008), Vol. 338.
47. D. Jimenez Rezende, S. Mohamed, "Variational inference with normalizing flows" in *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning,* F. Bach, D. Blei, Eds. (JMLR.org, 2015), **Vol. 37**.
48. J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah, Julia: A fresh approach to numerical computing. *SIAM Rev.* **59**, 65–98 (2017).
49. W. E, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.* **5**, 349–380 (2017).
50. J. Han, W. E, Deep learning approximation for stochastic control problems. arXiv:1611.07422 (2 November 2016).
51. J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci. U.S.A.* **115**, 8505–8510 (2018).
52. J. Sirignano, K. Spiliopoulos, Dgm: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **375**, 1339–1364 (2018).
53. W. Li, E. K. Ryu, S. Osher, W. Yin, W. Gangbo, A parallel method for earth mover's distance. *J. Sci. Comput.* **75**, 182–197 (2017).
54. E. Haber, R. Horesh, A multilevel method for the solution of time dependent optimal transport. *Numer. Math. Theory Methods Appl.* **8**, 97–111 (2015).
55. P. Gabriel, M. Cuturi, Computational optimal transport. *Found. Trends Mach. Learn.* **11**, 355–602 (2019).
56. W. Li, S. Osher, Constrained dynamical optimal transport and its Lagrangian formulation. arXiv:1807.00937 (11 September 2018).
57. L. Zhang, W. E, L. Wang, Monge-ampère flow for generative modeling. arXiv.org (26 September 2018).
58. L. Yang, G. E. Karniadakis, Potential flow generator with L_2 optimal transport regularity for generative models optimal transport regularity for generative models. arXiv.org (29 August 2019).
59. J. Lin, K. Lensink, E. Haber, Fluid flow mass transport for generative networks. ariXiv:1910.01694 (7 October 2019).
60. J. Yang, X. Ye, R. Trivedi, H. Xu, H. Zha, Learning deep mean field games for modeling large population behavior. arXiv:1711.03156 (22 April 2018).
61. R. Carmona, M. Laurière, Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: II – The finite horizon case. arXiv:1908.01613 (5 August 2019).
62. R. Carmona, M. Laurière, Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: I – The ergodic case Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: I – The ergodic case. arXiv:1907.05980 (13 July 2019).
63. A. Mang, L. Ruthotto, A Lagrangian gauss-Newton-Krylov solver for mass- and intensity-preserving diffeomorphic image registration. *SIAM J. Sci. Comput.* **39**, B860–B885 (2017).
64. J. Nocedal, S. Wright, *Numerical Optimization* (Springer Series in Operations Research and Financial Engineering, Springer Science & Business Media, New York, NY, 2006).
65. R. Bellman, *Introduction to Matrix Analysis* (Society for Industrial and Applied Mathematics, Philadelphia, PA, ed. 2, 1997).
66. L. Ambrosio, N. Gigli, G. Savaré, *Gradient Flows in Metric Spaces and in the Space of Probability Measures* (Lectures in Mathematics ETH Zürich, Birkhäuser Verlag, Basel, Switzerland, ed. 2, 2008).
67. H. Robbins, S. Monro, A stochastic approximation method. *Ann. Math. Stat.* **22**, 400–407 (1951).
68. A. Nemirovski, A. Juditsky, G. Lan, A. Shapiro, Robust stochastic approximation approach to stochastic programming. *SIAM J. Optim.* **19**, 1574–1609 (2009).
69. L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning. *SIAM Rev.* **60**, 223–311 (2018).
70. H. Li, Z. Xu, G. Taylor, T. Goldstein, "Visualizing the loss landscape of neural nets" in *Thirty-second Annual Conference on Neural Information Processing Systems 2018*, S. Bengio *et al.*, Eds. https://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf. Accessed 3 April 2020.
71. K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition" *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition "Deep residual learning for image recognition*, G. Hua, H. Jégou, Eds. (Institute of Electrical and Electronics Engineers, 2016), pp. 770–778.

72. W. E, A proposal on machine learning via dynamical systems. *Commun. Math. Stat.* **5**, 1–11 (2017).
73. E. Haber, L. Ruthotto, Stable architectures for deep neural networks. *Inverse Probl.* **34**, 1–22 (2017).
74. Q. Li, L. Chen, C. Tai, W. E. Maximum principle based algorithms for deep learning. arXiv:1710.09513 (2 June 2018).
75. B. Chang, L. Meng, E. Haber, F. Tung, D. Begert, "Multi-level residual networks from dynamical systems view" in *International Conference on Learning Representations.* https://openreview.net/pdf?id=SyJS-OgR-. Accessed 3 April 2020.
76. S. Günther, L. Ruthotto, J. B. Schroder, E. C. Cyr, N. R. Gauger, Layer-parallel training of deep residual neural networks. *SIAM J. Math. Data Sci.* **2**, 1–23 (2019).
77. M. Innes, Flux: Elegant machine learning with Julia. *J. Open Source Software* **3**, 602 (2018).
78. L. V. Kantorovich, On a problem of Monge. *J. Math. Sci.* **133**, 1383–1383 (2006).
79. L. C. Evans, "Partial differential equations and Monge-Kantorovich mass transfer" in *Current Developments in Mathematics*, D. Jerison *et al.*, Eds. (International Press of Boston, 1997), pp. 65–126.
80. L. Ambrosio, "Lecture notes on optimal transport problems" in *Mathematical Aspects of Evolving Interfaces*, P. Colli, Ed. (Springer, Berlin, Germany, 2003), pp. 1–52.
81. C. Villani, *Topics in Optimal Transportation* (American Mathematical Society, 2003).