Abstract. We propose structure-preserving neural-network-based numerical schemes to solve both $L^2$-gradient flows and generalized diffusions. In more detail, by using neural networks as tools for spatial discretization, we introduce a structure-preserving Eulerian algorithm to solve $L^2$-gradient flows and a structure-preserving Lagrangian algorithm to solve generalized diffusions. The Lagrangian algorithm for a generalized diffusion evolves the “flow map” which determines the dynamics of the generalized diffusion. This avoids the non-trivial task of computing the Wasserstein distance between two probability functions.

Unlike most existing methods that construct numerical discretizations based on the strong or weak form of the underlying PDE, the proposed schemes are constructed using variational formulations of these PDEs for preserving their variational structures. Our schemes first perform temporal discretization on these variational systems. By doing so, they are very computer-memory-efficient. Moreover, instead of directly solving the obtained nonlinear systems after temporal and spatial discretization, the minimizing movement scheme is utilized to evolve the solutions. This guarantees the monotonic decay of the energy of the system, and is crucial for the long-term stability of numerical computation. Lastly, the proposed neural-network-based scheme is mesh-free, and enables us to solve gradient flows in high dimensions. Various numerical experiments are presented to demonstrate the accuracy and energy stability of the proposed numerical approaches.

1. Introduction. Neural networks (NNs) have shown remarkable success across diverse scientific disciplines [12, 29, 38, 40, 44]. Due to the strong expressive power of neural network (NN) [3, 10, 20], and in particular deep neural network (DNN), there are substantial interests in solving partial differential equations (PDEs) by developing neural-network-based algorithms, especially for PDEs in high dimensions, in which traditional numerical algorithms, such as finite difference, finite element, and finite volume methods, suffer from the well-known curse of dimensionality (CoD) [25]. Namely, the computational complexity of the numerical algorithm increases exponentially as a function of the dimensionality of the problem [4, 25].

Ideas of solving differential equations numerically with a NN discretization have been explored for a couple of decades [13, 21, 34, 35, 39, 57, 62, 64, 70]. Examples include physics-informed neural network (PINN) [57], deep Ritz method (DRM) [21], deep Galerkin method (DGM) [62], variational PINN [34], and weak adversarial network (WAN) [76] to name a few. A key component of these aforementioned approaches is to represent solutions of PDEs via NNs. This makes the approximation function to belong to a space of nonlinear functions, which may lead to a robust estimation by sparser representation and cheaper computation [34].

Very recently, efforts have been made in using NNs to solve evolution PDEs [8, 16]. Instead of approximating the solution of the evolution PDE over the whole time-space domain, these approaches use NNs with time-dependent parameters to represent the solutions. So the algorithms update the parameters sequentially from one time slice to another, similar to that of time marching numerical methods. However, most
of these methods are developed based on the strong or weak forms of the evolution equations, which may fail to preserve the underlying variational structures of the original evolution PDEs. As a consequence, a significantly small time step size may be needed to guarantee the stability of the algorithm.

The goal of this paper is to develop structure-preserving, energetic variational neural network (EVNN) discretizations to various gradient flows or evolution PDEs that can be derived by using the Energetic Variational Approach (EnVarA) [23, 45, 67]. More specifically, we consider the following two types of gradient flows modeled by the EnVarA:

- **An $L^2$-gradient flow** that satisfies an energy-dissipation law

$$\frac{d}{dt} F[\varphi] = -\int_{\Omega} \eta |\varphi_t|^2 dx.$$  

Here $\varphi$ is the state variable, and the functional $F[\varphi] = \int_{\Omega} W(\varphi, \nabla \varphi, \ldots)dx$ is the free energy of the system with $W(\varphi, \nabla \varphi, \ldots)$ being the free energy density. $\eta > 0$ is the dissipation rate that determines how the system evolves to the equilibrium. $\frac{1}{\eta}$ is known as the mobility. $\eta$ can depend on both $x$ and $\varphi$ in general. The corresponding $L^2$-gradient flow equation can be written as

$$\eta \varphi_t = -\frac{\delta F}{\delta \varphi}.$$  

- **A generalized diffusion** that satisfies an energy-dissipation law

$$\frac{d}{dt} F[\rho] = -\int_{\Omega} \eta(\rho) |u|^2 dx,$$

where $F[\rho]$ is the free energy, $\eta(\rho) > 0$ is the friction coefficient, and $\rho$ is a conserved quantity satisfying the physics law of conservation of mass

$$\rho_t + \nabla \cdot (\rho u) = 0,$$

with $u$ being the “velocity”. The corresponding PDE is given by

$$\rho_t = \nabla \cdot \left( \frac{\rho^2}{\eta(\rho)} \nabla \mu \right), \quad \mu = \frac{\delta F}{\delta \rho}.$$  

The detailed variational procedures that derive the underlying PDEs from these prescribed energy-dissipation laws are described in Sections 2.1.1 and 2.1.2 of the paper.

A key idea of our approach is to directly construct numerical discretizations based on the continuous energy-dissipation laws (1.1) and (1.3), respectively. The proposed schemes have advantages of preserving a discrete counterpart of the energy dissipation property of the system in numerical solution, which is crucial for long term stability of numerical computation. So the schemes are termed as structure-preserving schemes. More specifically, we propose a structure-preserving Eulerian algorithm to solve the $L^2$-gradient flows and a structure-preserving Lagrangian algorithm to solve the generalized diffusions. Both algorithms utilize the variational formulation of the gradient flows and the minimizing movement scheme [24] for temporal discretization to ensure the monotonic decay of the energy over time. Moreover, our approach firstly constructs the temporal discretization, and then the spatial discretization. The advantage of doing so is that our method is efficient in utilizing computer memory in neural-network-based implementations.
We note that the numerical algorithm to solve the generalized diffusion (1.5) can also be formulated in the Eulerian frame of reference based on the notion of Wasserstein gradient flow [33]. However, it is often challenging to compute the Wasserstein distance between two probability densities efficiently in high accuracy, which often requires to solve an additional min-max problem or minimization problem [5, 6, 9, 31]. We refer the interested readers to a recent work [31] for a neural network implementation of the minimizing movement scheme with the Wasserstein distance.

Since gradient flows also provide a continuous variational formulation of many machine learning algorithms [19, 66], the numerical approaches developed here will have wide applications in many machine learning tasks, such as supervised learning [19], variational inference [66], density estimation [63] and generative models [30].

The rest of the paper is organized as follows. Section 2 reviews the EnVarA and some existing neural-network-based numerical approaches for solving gradient flows. Section 3 of the paper is devoted to derivation of the proposed EVNN schemes. Numerical simulation results are described in Section 4. Conclusions are drawn in Section 5.

2. Preliminary.

2.1. Energetic Variational Approach. In this section, we briefly review the EnVarA formulation of both $L^2$-gradient flows and generalized diffusions. Motivated by the non-equilibrium thermodynamics, especially the seminal works of Rayleigh [58] and Onsager [53, 54], the EnVarA describes an isothermal and closed system by its energy-dissipation law

$$\frac{d}{dt}E_{\text{total}} = -\nabla \leq 0,$$

where $E_{\text{total}}$ is the sum of the kinetic energy $K$ and the Helmholtz free energy $F$ of the system, and $\nabla$ is the rate of energy dissipation, which in fact is the rate of entropy production. Eq. (2.1) can be easily derived via the combination of the first and second laws of thermodynamics. The EnVarA has been employed to develop thermodynamically consistent models for various complex systems, such as two-phase flows [72, 73, 75], liquid crystals [45], ionic solutions [22], and reactive fluids models [68, 69]. We refer the interested readers to [23, 67] for a comprehensive review of the EnVarA.

In contrast to Newton’s vectorial mechanics, in which momentum and force are considered, the variational approach mainly concerns with two fundamental scalar quantities, the kinetic energy $K$ and the free energy $F$, and their dissipation over time $\nabla$. After defining the energy-dissipation law, the EnVarA derives the dynamics of the systems through the Least Action Principle (LAP) and the Maximum Dissipation Principle (MDP) in combination of kinematic relations (assumptions) of the physical variables of the system. The LAP states that the equation of motion of a Hamiltonian system can be derived from the variation of the action functional $A = \int_{t_0}^{t_f} (K - F) dt$ with respect to the flow map $x(X, t)$ (the trajectory of material points in the Lagrangian coordinates $X$ if applicable). This gives rise to a unique procedure to derive the “conservative” (or Hamiltonian) part of the force of the system. Here $x$ is the Eulerian coordinates, and $X$ is the Lagrangian coordinates or initial configuration of the system. See Section 2.1.2 for details. The MDP, in which variation of the dissipation potential $D$ that equals to $\frac{1}{2} \nabla$ in the linear response regime (near equilibrium) with respect to $x_t$, i.e., the velocity, gives the “dissipative” part of the force of the system.
In turn, the force balance condition
\[ \frac{\delta D}{\delta x_t} = \frac{\delta A}{\delta x} \]
leads to the evolution equation (equation of motion) of the system. The thermodynamic consistency of the derived evolution equations, which is guaranteed by the variational procedures described here, is also crucial for establishing well-posedness of the underlying system.

The variational principle are invariant with respect to the choice of the reference frame for the system, and assure the thermodynamic consistency of the system. Additionally, Ritz-type methods can be readily employed to find the solution to the system without the need of working with the corresponding PDEs [46, 65, 74].

Throughout this paper, we consider the cases in which \( K = 0 \), then the system can be viewed as a generalized gradient flow.

2.1.1. \( L^2 \)-gradient flow. \( L^2 \)-gradient flows are those systems satisfying the energy-dissipation law:
\[ \frac{d}{dt} F[\varphi] = - \int_{\Omega} \eta |\varphi_t|^2 dx . \]
where \( \varphi \) is the state variable, \( F[\varphi] \) is the Helmholtz free energy, and \( \eta > 0 \) is the dissipation rate. We assume that \( \varphi \) is a scalar function throughout this paper. One can also view \( \varphi \) as the generalized coordinates of the system [14]. By treating \( \varphi \) as \( x \), the general variational procedure (2.2) leads to the \( L^2 \)-gradient flow equation (1.2):
\[ \eta \varphi_t = - \frac{\delta F}{\delta \varphi} . \]
Many problems arising in soft matter physics, material science and machine learning can be modeled as \( L^2 \)-gradient flows. Examples include Allen-Cahn equation [15], Oseen-Frank and Landau-de Gennes models of liquid crystals [11, 42], phase field crystal models of quasicrystal [32, 41], and generalized Ohta-Kawasaki model of diblock and triblock copolymers [52]. For example, by taking \( F[\varphi] = \int_{\Omega} \frac{1}{2} |\nabla \varphi|^2 + \frac{1}{4\epsilon^2} (\varphi^2 - 1)^2 dx \), the classical Ginzburg-Landau free energy and letting \( \eta(\varphi) = 1 \), one gets the Allen-Cahn equation \( \varphi_t = \Delta \varphi - \frac{1}{\epsilon^2} (\varphi^2 - 1) \varphi \).

2.1.2. Generalized diffusion. Another type of gradient flows is known as generalized diffusions, which describe the space-time evolution of a conserved quantity \( \rho(x, t) \). Due to the physics law of mass conservation, \( \rho(x, t) \) satisfies the kinematics relation (1.4), \( \partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0 \), where \( \mathbf{u} \) is an averaged velocity. Given the velocity field \( \mathbf{u}(x, t) \), one can define a flow map \( x(X, t) \) through
\[
\begin{align*}
\frac{d}{dt} x(X, t) &= u(x(X, t), t) , \\
x(x, 0) &= X ,
\end{align*}
\]
where \( X \in \Omega_0 \) is the Lagrangian coordinates and \( x \in \Omega_t \) is the Eulerian coordinates. Here \( \Omega_0 \) is the initial configuration of the system, and \( \Omega_t \) is the configuration of the system at time \( t \). For a fixed \( X \), \( x(X, t) \) describes the trajectory of a particle (or a material point) labeled by \( X \); while for a fixed \( t \), \( x(X, t) \) is a diffeomorphism from
The existence of the flow map $\mathbf{x}(\mathbf{X}, t)$ requires a certain regularity of $u(\mathbf{x}, t)$, for instance, being Lipschitz in $\mathbf{x}$.

In a Lagrangian picture, evolution of the density function $\rho(\mathbf{x}, t)$ is determined by the evolution of the $\mathbf{x}(\mathbf{X}, t)$ through the kinematics relation (1.4) (written in the Lagrangian frame of reference). More precisely, one can define the deformation tensor associated with the flow map $\mathbf{x}(\mathbf{X}, t)$ by

\begin{equation}
\tilde{F}(\mathbf{x}(\mathbf{X}, t), t) = F(\mathbf{X}, t) = \nabla_{\mathbf{X}} \mathbf{x}(\mathbf{X}, t).
\end{equation}

Without ambiguity, we do not distinguish $F$ and $\tilde{F}$ in the rest of the paper. Let $\rho_0(\mathbf{X})$ be the initial density, then the law of mass conservation which states that

\begin{equation}
\int_{\Omega_t} \rho(\mathbf{x}, t) d\mathbf{x} = \int_{\Omega_0} \rho_0(\mathbf{X}) d\mathbf{X},
\end{equation}

$\rho(\mathbf{x}, t)$ indicates

\begin{equation}
\rho(\mathbf{x}(\mathbf{X}, t), t) = \frac{\rho_0(\mathbf{X})}{\det F(\mathbf{X}, t)}
\end{equation}
in the Lagrangian frame of reference. This is equivalent to $\rho_t + \nabla \cdot (\rho \mathbf{u}) = 0$ in the Eulerian frame.

In the EnVarA framework, a generalized diffusion can be described by an energy-dissipation law (1.3):

\begin{equation}
\frac{d}{dt} F[\rho] = - \int \eta(\rho) |\mathbf{u}|^2 d\mathbf{x}.
\end{equation}

A typical form of the free energy $F[\rho]$ is

\begin{equation}
F[\rho] = \int_{\Omega} \omega(\rho) + \rho V(\mathbf{x}) + \frac{1}{2} \int_{\Omega} K(\mathbf{x}, \mathbf{y}) \rho(\mathbf{x}) \rho(\mathbf{y}) d\mathbf{y} d\mathbf{x},
\end{equation}

where $V(\mathbf{x})$ is a potential field, and $K(\mathbf{x}, \mathbf{y})$ is a symmetric non-local interaction kernel. By a direct computation (see Appendix for details), the force balance equation (2.2) can be written as

\begin{equation}
\eta(\rho) \mathbf{u} = - \rho \nabla \mu.
\end{equation}

Combining (2.6) with the kinematics equation (1.4), one obtains a generalized diffusion equation

\begin{equation}
\rho_t = \nabla \cdot \left( \frac{\rho^2}{\eta(\rho)} \nabla \mu \right).
\end{equation}

The velocity equation (2.6) can be viewed as the equation of the flow map $\mathbf{x}(\mathbf{X}, t)$ in the Lagrangian coordinates

\begin{equation}
\eta \left( \frac{\rho_0(\mathbf{X})}{\det F(\mathbf{X})} \right) \frac{d}{dt} \mathbf{x}(\mathbf{X}, t) = - \frac{\rho_0(\mathbf{X})}{\det F(\mathbf{X})} \mathbf{F}^{-T} \nabla_{\mathbf{X}} \mu \left( \frac{\rho_0(\mathbf{X})}{\det F(\mathbf{X})} \right),
\end{equation}

5
where $F(X) = \nabla X x(X, t)$. The flow map (2.8) is a highly nonlinear equation of $x(X, t)$ that involves both $F$ and $\det F$. It is rather unclear how a suitable spatial discretization to Eq. (2.8) would be introduced. However, the generalized diffusion can be viewed as a $L^2$-gradient flow of the flow map in the space of diffeomorphism. This gives rise to a natural discretization to the generalized diffusion [46].

**Remark 2.1.** In the case that $\eta(\rho) = \rho$, the generalized diffusions can be viewed as Wasserstein gradient flows in space of all probability densities having finite second moments $P_2(\Omega)$ [33]. Formally, the Wasserstein gradient flow can be defined as a continuous time limit ($\tau \to 0$) from the semi-discrete scheme, known as the JKO scheme,

$$
\rho_{k+1} = \arg\min_{\rho \in P_2} \frac{1}{2\tau} W_2(\rho, \rho_k)^2 + \mathcal{F}[\rho], \quad k = 0, 1, 2, \ldots,
$$

where $P_2 = \left\{ \rho : \Omega \to [0, \infty) \mid \int_\Omega \rho d\mathcal{L} = 0, \int_\Omega |x|^2 \rho(x) d\mathcal{L} < \infty \right\}$ and $W_2(\rho, \rho_k)$ is the Wasserstein distance between $\rho$ and $\rho_k$. The Wasserstein gradient flow is an Eulerian description to these systems [6]. Other choices of dissipation can define other metrics in the space of probability measures [1, 43].

### 2.2. Neural-network-based numerical schemes for PDEs

In this subsection, we briefly review some existing approaches for solving PDEs by neural-network-based algorithms. We refer interested readers to [18, 48] for detailed reviews.

Consider a PDE subject to some boundary condition

$$
\mathcal{L} \varphi(x) = f(x), \quad x \in \Omega \subset \mathbb{R}^d, \quad \mathcal{B} \varphi(x) = g(x), \quad x \in \partial \Omega \subset \mathbb{R}^{d-1}.
$$

Assume the solution can be approximated by a neural network $\varphi_{NN}(x; \Theta)$, where $\Theta$ is the set of parameters used in the neural network. The loss function of the PINN and similar methods in [13, 57, 62, 70] can be defined by

$$
L(\Theta) = \frac{1}{N_{\text{in}}} \sum_{i=1}^{N_{\text{in}}} (\mathcal{L} \varphi_{NN}(x_i; \Theta) - f(x_i))^2 + \frac{1}{N_{\text{b}}} \sum_{j=1}^{N_{\text{b}}} (\mathcal{B} \varphi_{NN}(s_j; \Theta) - g(s_j))^2,
$$

where $\{x_i\}_{i=1}^{N_{\text{in}}}$ and $\{s_j\}_{j=1}^{N_{\text{b}}}$ are sets of samples in $\Omega$ and $\partial \Omega$, respectively. These samples can be drawn uniformly or by following some prescribed distributions. Minimizers of the loss function (2.11) can be obtained by using some optimization methods, such as the standard stochastic gradient descent (SGD) method. Of course, the objective function of this minimization problem in general is not convex even when the initial problem is. Obtaining the global minimum of (2.11) is in general non-trivial.

Unlike the PINN which adopts the strong forms of the PDEs, the deep Ritz method [21] is designed for solving problems using variational formulations (The Rayleigh–Ritz or Ritz method is to find approximate solutions for problems described using variational formulations). The loss function of the DRM is defined in a variational fashion [21]. If Eq. (2.10) is an Euler-Lagrangian equation of some functional

$$
I[\varphi, \nabla \varphi] = \int_\Omega W(\varphi(x), \nabla_x \varphi(x)) d\mathcal{L},
$$

the loss function can be defined directly by

$$
L(\Theta) = \frac{1}{N_{\text{in}}} \sum_{i=1}^{N_{\text{in}}} W(\varphi_{NN}(x_i), \nabla_x \varphi_{NN}(x_i)) + \frac{1}{N_{\text{b}}} \sum_{j=1}^{N_{\text{b}}} (\mathcal{B} \varphi_{NN}(s_j) - g(s_j))^2.
$$
Here the last term is a penalty for imposing the boundary condition, \( \{ x_i \}_{i=1}^{N_i} \) and \( \{ s_j \}_{j=1}^{N_b} \) can be uniformly sampled from \( \Omega \) and \( \partial \Omega \) or sampled following some other prescribed distributions, respectively. We also note that both the variational PINN [34] and the WAN [76] utilized the Galerkin formulation to solve PDEs. While the variational PINN [34] represents the solution via a DNN, and the test functions still belong to linear function spaces; the WAN [76] employs the primal and adversarial networks to parameterize the weak solution and test functions, respectively. Results from these works underscore importance of formulating PDEs as variational problems to achieve better performance.

For the evolution equation

\[
\begin{align*}
\partial_t \varphi(x,t) &= F(t, x, \varphi), \quad (x, t) \in \Omega \times (0, \infty), \\
\varphi(x, 0) &= \varphi_0(x), \quad x \in \Omega
\end{align*}
\]

with a suitable boundary condition. The neural Galerkin method, proposed in a recent work [8], parameterizes the solution as \( \varphi_h(x; \Theta(t)) \) with \( \Theta(t) \) being the NN parameters, and defines a loss function in terms of \( \Theta \) and \( \dot{\Theta} \) through a residual function, i.e.,

\[
J(\Theta, \eta) = \frac{1}{2} \int |\nabla \Theta \varphi_h \cdot \eta - F(x, t, \varphi_h(x; \Theta))|^2 \nu_\Theta(x) ,
\]

where \( \nu_\Theta(x) \) is a suitable measure which might depend on \( \Theta \). By taking variation of \( J(\Theta, \eta) \) with respect to \( \eta \), the neural Galerkin method arrives at an ODE of \( \Theta(t) \):

\[
M(\Theta) \dot{\Theta} = F(t, \Theta), \quad \Theta(0) = \Theta_0 ,
\]

where \( M(\Theta) = \int \nabla \Theta \varphi_h \otimes \nabla \Theta \varphi_h \nu_\Theta(x) \), and \( F(t, \Theta) = \int \nabla \Theta \varphi_h F(x, t, \varphi_h) \nu_\Theta(x) \). The ODE (2.16) can be solved by standard explicit or implicit time-marching schemes.

It is worth mentioning that most of the existing approaches are Eulerian methods. For certain types of PDEs, such as generalized diffusions, it might be difficult to preserve the physical constraints, such as positivity and conservation of mass of a probability function by the Eulerian methods.

3. Energetic Variational Neural Network. In this section, we present the proposed structure-preserving EVNN schemes for solving both \( L^2 \)-gradient flows and generalized diffusions in detail. Our key idea is to integrate neural-network-based spatial discretization with the structure-preserving variational integrator for these two types of gradient flows [46]. We note that Eulerian [51], Lagrangian [46, 47] and particle methods [66] have been traditionally used for spatial discretization. Since neural networks are a powerful parametric approximation to an unknown function, and one can use Monte Carlo type approaches to compute high-dimensional integrals appeared in the problems, here we choose to use neural network discretization so that our scheme is capable of solving gradient flows in high-dimension.

We also note that, due to the nonlinear nature of neural network approximation, it is not straightforward to replace traditional spatial discretization by neural networks.

In the following subsections, we’ll describe how to construct the neural-network-based discrete schemes for the gradient flows (1.2) and (1.5), respectively, in detail.

3.1. Discrete Eulerian EVNN scheme for \( L^2 \)-gradient flow: temporal-then-spatial approach. To develop structure-preserving variational scheme for gradient flows, the authors proposes a discrete energetic variational approach in [46, 74],
which first constructs a finite-dimensional approximation to a continuous energy-dissipation law by introducing a spatial discretization to the state variable $\varphi$, denoted by $\varphi_h(x; \Theta(t))$, where $\Theta(t) \in \mathbb{R}^K$ is the parameter to be determined. This results in a semi-discrete energy-dissipation law:

$$
\frac{d}{dt} \mathcal{F}_h[\Theta(t)] = -\Delta_h[\Theta(t), \Theta'(t)],
$$

where $\Theta'(t)$ is its time derivative. $\mathcal{F}_h[\Theta(t)] = \mathcal{F}[\varphi_h(x; \Theta)]$, and $\Delta_h[\Theta(t), \Theta'(t)] = \int \eta(\varphi_h)\nabla \varphi_h \cdot \Theta'(t)^2 \, dx$. For example, in Galerkin methods, one can let $\varphi_h(x, t)$ be $\varphi_h(x, t) = \sum_{i=1}^{K} \gamma_i(t) \psi_i(x)$ with $\{\psi_i\}_{i=1}^{K}$ being the set of basis functions. Then $\Theta(t)$ is a vector given by $\Theta(t) = (\gamma_1(t), \gamma_2(t), \ldots, \gamma_K(t))^T \in \mathbb{R}^K$.

By employing the energetic variational approach in the semi-discrete level (3.1), one can obtain an ODE system of $\Theta(t)$. Particularly, in the linear response regime, $\mathcal{D}_h[\Theta(t), \Theta'(t)] = \frac{1}{2} \int \eta(\varphi_h) |\partial_t \varphi_h|^2 \, dx$ is a quadratic function of $\Theta'$. The ODE system of $\Theta(t)$ can then be written as

$$
\mathcal{D}(\Theta) \Theta'(t) = -\frac{\delta \mathcal{F}_h}{\delta \Theta},
$$

where

$$
\frac{\delta \mathcal{F}_h}{\delta \Theta} = \frac{\delta \mathcal{F}}{\delta \varphi} \nabla \Theta \varphi_h, \quad \mathcal{D}(\Theta) = \int \eta(\varphi_h)(\nabla \Theta \varphi_h \otimes \nabla \Theta \varphi_h) \, dx.
$$

The ODE (3.2) is same to the ODE (2.16) in the neural Galerkin method [8], although the derivation is totally different.

Since (3.2) is a finite-dimensional gradient flow, one can then construct a minimizing movement scheme for $\Theta$: finding $\Theta^{n+1}$ such that

$$
\Theta^{n+1} = \arg \min_{\Theta \in S_{ad}^h} J_n(\Theta), \quad J_n(\Theta) = \frac{(\mathcal{D}_n(\Theta - \Theta^n)) \cdot (\Theta - \Theta^n)}{2\tau} + \mathcal{F}_h(\Theta),
$$

where $\mathcal{D}_n$ is a constant matrix. A typical choice of $\mathcal{D}_n = \mathcal{D}(\Theta^n)$. An advantage of this scheme is that

$$
\mathcal{F}_h(\Theta^{n+1}) \leq J_n^{n}(\Theta^{n+1}) \leq J_n^{n}(\Theta^n) = \mathcal{F}_h(\Theta^n),
$$

which guarantees the energy stability for the discrete free energy $\mathcal{F}_h(\Theta)$. Moreover, by choosing a proper optimization method, we can assure that $\varphi^{n+1}$ stays in the admissible set $S$.

Although neural networks can be used to construct $\varphi_h(x; \Theta(t))$, it might be expensive to compute $\nabla \Theta \varphi_h \otimes \nabla \Theta \varphi_h$ in $\mathcal{D}(\Theta)$. Moreover, $\mathcal{D}(\Theta)$ is not a sparse matrix, and requires a lot of computer memory to store when a very deep neural network is used.

To overcome these difficulties, we propose an alternative approach. Here we first introduce a temporal discretization. Let $\tau$ be the time step-size. For the $L^2$-gradient flow (1.2), given $\varphi^n$, which is a numerical solution at $t^n = n\tau$, one can obtain $\varphi^{n+1}$ by solving the following optimization problem: finding $\varphi^{n+1}$ in some admissible set $S$ such that

$$
\varphi^{n+1} = \arg \min_{\varphi \in \mathcal{S}} J^n(\varphi), \quad J^n(\varphi) = \frac{1}{2\tau} \int \eta(\varphi^n)|\varphi - \varphi^n|^2 \, dx + \mathcal{F}[\varphi].
$$
Let \( \varphi_h(x; \Theta) \) be a finite-dimensional approximation to \( \varphi \) with \( \Theta \in \mathbb{R}^K \) being the parameter of the spatial discretization (e.g., weights of linear combination in a finite element discretization) yet to be determined, then the minimizing movement scheme can be written in term of \( \Theta \): finding \( \Theta^{n+1} \) such that

\[
\Theta^{n+1} = \arg \min_{\Theta \in S_h} J^n_h(\Theta), \quad J^n_h(\Theta) = \frac{1}{2\tau} \int_{\Omega} \eta^n \left| \varphi_h(x; \Theta) - \varphi_h(x; \Theta^n) \right|^2 dx + F_h[\Theta].
\]

Here \( \Theta^n \) is the value of \( \Theta \) at time \( t^n \). \( F_h[\Theta] = F[\varphi_h(x; \Theta)] \), and \( \eta^n = \eta(\varphi_h(x; \Theta^n)) \).

**Remark 3.1.** The connection between the minimizing movement scheme (3.7) derived by temporal-discretization-first, and the minimizing movement scheme (3.4) derived by spatial-discretization-first, can be shown with a direct calculation. One can show that an optimal solution to the minimization problem (3.7) \( \Theta^{n+1} \) satisfies

\[
\frac{\delta J^n_h(\Theta)}{\delta \Theta} \bigg|_{\Theta^{n+1}} = \frac{1}{\tau} \int_{\Omega} \eta^n \left( \varphi_h(x; \Theta^{n+1}) - \varphi_h(x; \Theta^n) \right) \nabla \Theta \varphi_h \bigg|_{\Theta^{n+1}} dx + \frac{\delta F_h}{\delta \Theta} \bigg|_{\Theta^{n+1}} = 0
\]

for some \( \Theta^* \), where the second equality follows the mean value theorem. In the case of Galerkin methods, as \( \nabla \Theta \varphi_h \) is independent on \( \Theta \), \( \Theta^{n+1} \) is a solution to an implicit Euler scheme for the ODE (3.2) in which \( D(\Theta) \) is treated explicitly.

By choosing a certain neural network as an approximation to \( \varphi \), denoted by \( \varphi_{NN}(x; \Theta) \) (\( \Theta \) is used to denote all parameters in the neural network), the EVNN scheme for solving the \( L^2 \)-gradient flow is summarized in Alg. 3.1.

### Algorithm 3.1 Numerical Algorithm for solving the \( L^2 \)-gradient flow

For a given initial condition \( \varphi_0(x) \), compute \( \Theta^0 \) by solving

\[
\Theta^0 = \arg \min_{\Theta} \int_{\Omega} \left| \varphi_{NN}(x; \Theta) - \varphi_0(x) \right|^2 dx;
\]

At each step, update \( \Theta^{n+1} \) by solving the optimization problem

\[
\Theta^{n+1} = \arg \min_{\Theta} \left( \frac{1}{2\tau} \int_{\Omega} \eta^n \left| \varphi_{NN}(x; \Theta) - \varphi_{NN}(x, \Theta^n) \right|^2 dx + F[\varphi_{NN}(x; \Theta)] \right).
\]

We have \( \varphi_{NN}(x; \Theta^n) \) as a numerical solution at time \( t^n = n\tau \).

It can be noticed that both Eq. (3.9) and Eq. (3.10) involve integration in the computational domain \( \Omega \). This integration is often computed by using Monte-Carlo or Quasi Monte-Carlo algorithms [60].

**Remark 3.2.** It is straightforward to incorporate other variational high-order temporal discretizations to solve the gradient flows. For examples, a second-order accurate BDF2 scheme can be reformulated as an optimization problem

\[
\Theta^n = \arg \min_{\Theta} \left( \frac{\eta}{\tau} \int_{\Omega} \varphi_{NN}(x; \Theta) - \varphi_{NN}(x, \Theta^{n-1})|^2 dx \right.
\]

\[
- \frac{\eta}{4\tau} \int_{\Omega} \left| \varphi_{NN}(x; \Theta) - \varphi_{NN}(x, \Theta^{n-2}) \right|^2 dx + F[\varphi_{NN}(x; \Theta)] \right).
\]
A modified Crank-Nicolson time-marching scheme can be reformulated as

\[
\Theta^n = \arg \min_{\Theta} \left( \eta \frac{1}{2} \int_{\Omega} (\varphi_{NN}(x; \Theta) - \varphi_{NN}(x; \Theta^{n-1}))^2 \, dx + \mathcal{F}[\varphi_{NN}(x; \Theta)] + (\nabla_{\Theta} \mathcal{F}[\varphi_{NN}(x; \Theta^{n-1})], \Theta - \Theta^{n-1}) \right). 
\]

Here we assume that \( \eta \) is a constant for simplicity.

### 3.2. Discrete Lagrangian EVNN scheme for generalized diffusions.

In this subsection, we show how to formulate an EVNN scheme in the Lagrangian frame of reference for generalized diffusions. As discussed previously, a generalized diffusion can be viewed as an \( L^2 \)-gradient flow of the flow map \( \rho(X, t) \) in the space of diffeomorphism. Hence, the EVNN scheme for the generalized diffusion can be formulated in terms of a minimizing movement scheme for the flow map given by

\[
\Phi^{n+1} = \arg \min_{\Phi \in \text{Diff}} \frac{1}{2\tau} \int |\Phi(X) - \Phi^n(X)|^2 \rho_0(X) \, dX + \mathcal{F}[\rho_0 \delta(\Phi^{-1}(x))],
\]

where \( \Phi^n(X) \) is a numerical approximation of the flow map \( \rho(X, t) \) at \( t^n = n\tau \), \( \rho_0(X) \) is the initial density. \( \mathcal{F}[\rho] \) is the free energy for the generalized diffusion defined in Eq. (2.5), and

\[
\rho_0 \delta(\Phi^{-1}(x)) := \frac{\rho_0(\Phi^{-1}(x))}{\det F(\Phi^{-1}(x))}, \quad \text{Diff} = \{ \Phi : \mathbb{R}^d \to \mathbb{R}^d \ | \ \Phi \text{ is a diffeomorphism} \}.
\]

One can then parameterize \( \Phi : \mathbb{R}^d \to \mathbb{R}^d \) by a suitable neural network. The remaining procedure is almost the same as that of the last subsection.

The optimization problem (3.13) seeks for a transport map between \( \rho^0 \) and \( \rho^{n+1} \). Here \( \rho^n \) stands for a numerical approximation to \( \rho(x, t) \) at \( t^n = n\tau \). Notice that \( \rho^n(x) = \rho^0 \delta(\Phi^n)^{-1}(x) \). As a consequence, it is often difficult to solve this optimization problem, and the minimizer to (3.13) may not even exist after a finite dimensional approximation to the space of diffeomorphism. To overcome this difficulty, at each time step, instead of seeking for an optimal \( \Phi^{n+1} \) between \( \rho^0 \) and \( \rho^{n+1} \), we seek for an optimal \( \Psi^{n+1} \) between \( \rho^n \) and \( \rho^{n+1} \). More precisely, we assume that

\[
\Phi^{n+1} = \Psi^{n+1} \circ \Psi^{n} \circ \Psi^{n-1} \ldots \circ \Psi^1.
\]

Given \( \rho^n \), one can compute \( \Psi^{n+1} \) by solving the following optimization problem

\[
\Psi^{n+1} = \arg \min_{\Psi \in \text{Diff}} \frac{1}{2\tau} \int |\Psi(x) - x|^2 \rho^n(x) \, dx + \mathcal{F}[\rho^n \delta(\Psi^{-1}(x))].
\]

The corresponding \( \rho^{n+1} \) can then be computed through \( \rho^{n+1}(x) = \rho^n \delta(\Psi^{n+1})^{-1}(x) \).

**Remark 3.3.** The scheme (3.14) can be viewed as a Lagrangian realization of the JKO scheme (2.9) for the Wasserstein gradient flow under some suitable assumption, although it is developed based on the \( L^2 \)-gradient flow structure in the space of diffeomorphism. According to the Benamou–Brenier formulation [5], the Wasserstein distance between two probability densities \( \rho_1 \) and \( \rho_2 \) can be computed by solving the optimization problem

\[
\mathbb{W}_2(\rho_1, \rho_2)^2 = \min_{(\rho, u) \in S} \int_0^1 \int |u|^2 \, dx \, dt,
\]

where

\[
\mathbb{W}_2(\rho_1, \rho_2)^2 = \min_{(\rho, u) \in S} \int_0^1 \int |u|^2 \, dx \, dt.
\]
where the admissible set of \((\rho, u)\) is given by

\[
S = \{ (\rho, u) \mid \rho_t + \nabla \cdot (\rho u) = 0, \quad \rho(x, 0) = \rho_1, \quad \rho(x, 1) = \rho_2 \}.
\]

If we further assume that \(u(x, t)\) is independent of \(t\), which is reasonable if \(\rho_1\) and \(\rho_2\) are close enough, we have \(u(x) = \Psi(x) - x\), where \(\Psi(x)\) is a transport map between \(\rho_1\) and \(\rho_2\), i.e., \(\rho_2(\Psi(x)) = \rho_1(x)/\det(\nabla_x \Psi)\). Then optimization problem (3.15) can be reformulated in a Lagrangian form, i.e.,

\[
W_2(\rho_1, \rho_2)^2 = \arg \min_{\Psi \in S_\Psi} \int \rho_1 |\Psi(x) - x|^2 \, dx,
\]

where

\[
S_\Psi = \left\{ \Psi : \mathbb{R}^d \to \mathbb{R}^d \mid \rho_2(\Psi(x)) = \frac{\rho_1(x)}{\det(\nabla_x \Psi)} \right\}.
\]

**Remark 3.4.** If \(\eta(\rho) \neq \rho\), we can formulate the optimization problem (3.14) as

\[
\Psi^{n+1} = \arg \min_{\Psi \in S_\Psi} \frac{1}{2\tau} \int |\Psi(x) - x|^2 \eta(\rho^n(x)) \, dx + F[\rho^n \circ \Psi^{-1}(x)].
\]

A subtle fact is that \(\det F^n = 1\) since we always start with an identity map.

The numerical algorithm for solving the generalized diffusion is summarized in Alg. 3.2.

**Algorithm 3.2** Numerical Algorithm for solving the generalized diffusion

- Given \(\{x^n_i\}_{i=1}^N\) and the densities \(\rho^n_i\) at \(t = n\tau\) and \(x^n_i\).
- Find \(\Psi^{n+1}(x) : \mathbb{R}^d \to \mathbb{R}^d\), by solving the optimization problem (3.14). To guarantee the energy stability, we should take \(\Psi\) as an approximation to an identity map initially when solving the optimization problem (3.14).
- After obtaining \(\Psi^{n+1}\), one updates \(\{x^{n+1}_i\}_{i=1}^N\) and \(\rho^{n+1}\) by

\[
x^{n+1}_i = \Psi^{n+1}(x^n_i), \quad \rho^{n+1}_i = \frac{\rho^n_i}{\det(\nabla \Psi^{n+1}(x^n_i))}.
\]

The next question is how to evaluate the numerical integration in (3.14), which requires to draw a set of samples. In principle, given \(\Phi^n\) and \(\rho_0(X)\), we can compute \(\rho^n(x)\) point-wisely through \(\rho^n(x) = \rho_0 \circ \Phi^n^{-1}(x)\). However, the computation might be expensive as it requires an efficient and accurate way to compute \((\Phi^n)^{-1}\). Alternatively, we can use the idea of the Lagrangian method [46] which only updates the value of \(\rho^n\) at the set of transformed samples \(\{\Phi^n(x^n_i)\}_{i=1}^N\) through

\[
\rho^n(\Phi^n(x^n_i)) = \frac{\rho_0(x^n_i)}{\det(\nabla \Phi^n(x^n_i))},
\]

where \(\{x^n_i\}_{i=1}^N\) is the set of initial samples. Let \(x^n_i = \Phi^n(x^n_i)\). Then for the general free energy (2.5), the loss function in Eq. (3.14) can be computed through

\[
J(\Psi) = \frac{1}{2\tau} \sum_{i=1}^N \rho^n_i \|\Psi(x^n_i) - x^n_i\|^2 |\Omega^n_i| + \sum_{i=1}^N \left( \omega \left( \frac{\rho^n_i}{\det(\nabla \Psi(x^n_i))} \right) \det(\nabla \Psi(x^n_i)) \right) |\Omega^n_i| + \frac{1}{2} \sum_{i=1}^N V(\Psi(x^n_i)) \rho^n_i |\Omega^n_i| + \frac{1}{2} \sum_{i,j=1}^N K(\Psi(x_i), \Psi(x_j)) \rho^n_i \rho^n_j |\Omega^n_i| |\Omega^n_j|,
\]

where

\[
\omega(x) = \begin{cases} 1 & \text{if } x \in S, \\ 0 & \text{if } x \notin S, \end{cases}
\]

and

\[
|\Omega^n_i| = \det(\nabla \Psi(x^n_i))^{-1}.
\]
where \(|\Omega^n_i|\) is the volume of the Voronoi cells associated with the set of points \(\{x^n_i\}\). \(\rho^n_i\) stands for \(\rho^n(x^n_i)\). Here the numerical integration is computed through a piece-wisely constant reconstruction of \(\rho^n\) based on its values at \(\{\Phi^n(x^n_i)\}_{i=1}^N\). In practice, if the initial samples are uniformly drawn in \(\Omega_0\), we take \(|\Omega^n_i| = \frac{|\Omega|}{N}\), although it might be a very rough approximation as \(x^n_i\) may not follow a uniform distribution in \(\Omega^n\). If the initial samples are drawn from \(\rho_0\), one can roughly assume that \(\{x^n_i\}\) follows the distribution \(\rho^n\), then according to the Monte-Carlo approach, the numerical integration can be evaluated as

\[
\begin{align*}
J(\Psi) &= \frac{1}{2\pi} \left( \frac{1}{N} \sum_{i=1}^{N} \|\Psi(x^n_i) - x^n_i\|^2 \right) \\
&+ \frac{1}{N} \sum_{i=1}^{N} \left( f_\omega \left( \frac{\rho^n_i}{\text{det}(\nabla\Psi(x^n_i))} \right) + V(\Psi(x^n_i)) \right) + \frac{1}{2\pi^2} \sum_{i,j=1}^{N} K(\Psi(x_i), \Psi(x_j)) ,
\end{align*}
\]

where \(f_\omega(\rho) = \omega(\rho)/\rho\). The proposed numerical method can be further improved if one can evaluate the integration more accurately. We will explore this in the future work.

The remaining question is how to parameterize a diffeomorphism using neural networks. This is discussed in detail in the next subsection.

### 3.3. Neural network architectures

In principle, the proposed numerical framework is independent of the choice of neural network architectures. However, different neural network architectures may lead to different numerical performance, arising from balance of approximation (representation power), optimization and generalization. In this subsection, we briefly discuss several neural network architectures that we use in the numerical experiments.

#### 3.3.1. Neural network architectures for Eulerian methods

For the Eulerian methods, one can construct a neural network to approximate the unknown function \(f : \mathbb{R}^d \rightarrow \mathbb{R}\). Shallow neural networks (two-layer neural networks) approximate \(f\) by functions of the form

\[
f(x; \Theta) = \sum_{i=1}^{N} \alpha_i \sigma(\omega_i \cdot x + b_i) = \alpha \cdot \sigma(Wx + b) ,
\]

where \(\sigma(\cdot) : \mathbb{R} \rightarrow \mathbb{R}\) is a fixed nonlinear activation function, \(N\) is the number of hidden nodes (neurons), and \(\Theta = \{\alpha_i, \omega_i, b_i\}_{i=1}^{N}\) are the NN parameters to be identified. Typical choices of activation functions include ReLU \(\sigma(x) = \max(x, 0)\) and the sigmoid function \(\sigma(x) = 1/(1 + e^{-2x})\).

A DNN can be viewed as a network composing of many hidden layers. More precisely, a DNN with \(L\) hidden layers represents a function \(f : \mathbb{R}^d \rightarrow \mathbb{R}\) by \cite{61}

\[
f(x; \Theta) = g \circ T^L \circ T^{L-1} \circ \ldots \circ T^{(1)}(x) ,
\]

where \(g(z) = \sum_{i=1}^{N_L} \gamma_i z_i\) is a linear map from \(\mathbb{R}^{N_L}\) to \(\mathbb{R}\), \(T^{(l)}\) is a nonlinear map from \(\mathbb{R}^{N_{l-1}}\) to \(\mathbb{R}^{N_l}\), and \(N_0 = d\). The nonlinear map \(T^{(l)}\) takes the form

\[
T^{(l)}(x_{l-1}) = \sigma(W_l x_{l-1} + b_l) ,
\]

where \(W_l \in \mathbb{R}^{N_{l-1} \times N_l}\), \(b_l \in \mathbb{R}^l\) and \(\sigma(\cdot)\) is a nonlinear activation function that acts component-wisely on vector-valued inputs.
Another widely used class of DNN model is residual neural network (ResNet). An L-block ResNet approximates an unknown function \( f(x) : \mathbb{R}^d \to \mathbb{R} \) by
\[
(3.26) \quad f_L(x; \Theta) = g(z_L(x)),
\]
where \( g(z) = \sum_{i=1}^{N} \gamma_i z_i \) is a linear map from \( \mathbb{R}^N \to \mathbb{R} \) and \( z_L(x) : \mathbb{R}^d \to \mathbb{R}^N \) is a nonlinear map defined through
\[
(3.27) \quad z_0 = V x, \quad z_i = \sigma_2(\alpha_i \sigma_1(W_i z_{i-1} + b_i) + z_{i-1}), \quad i = 1, 2, \ldots, L.
\]
Here \( V \in \mathbb{R}^{N \times d}, W_i \in \mathbb{R}^{M \times N}, b_i \in \mathbb{R}^M, \alpha_i \in \mathbb{R}^{N \times M} \) and \( \sigma_i(\cdot) \) is an element-wise activation function. The model parameters are \( \Theta = \{ \gamma, \alpha_i, W_i, b_i, V \} \). The original ResNet [27] takes \( \sigma_2 \) as a nonlinear activation function such as ReLU. Later studies indicate that one can also take \( \sigma_2 \) as the identity function [28, 18]. Then at an infinite length, i.e., \( L \to \infty \), (3.27) corresponds to the ODE
\[
(3.28) \quad \frac{dz}{dt} = f(z), \quad z_0 = x.
\]
Compared with fully connected neural network models which may suffer from numerical instabilities in the form of exploding or vanishing gradients [19, 26, 37], very deep ResNet can be constructed to avoid these issues.

In the current study, we use ResNet for the Eulerian EVNN method for the \( L^2 \)-gradient flows. The detailed neural network settings for each numerical experiment are described in the next section. It is worth mentioning that since the proposed numerical scheme uses neural networks with time-dependent parameters to approximate the solution to a gradient flow, it is not necessary to use a very deep neural network.

3.3.2. Neural network architectures for Lagrangian methods. The goal of the proposed Lagrangian method is to use a neural network to approximate a diffeomorphism from \( \mathbb{R}^d \) to \( \mathbb{R}^d \). There are two main difficulties in doing so. One is to guarantee that the map is diffeomorphism; the other is to compute the deformation gradient tensor \( F \) and its determinant \( \det F \) efficiently and in a robust manner. Fortunately, various neural network architectures have been proposed to approximate a transport map. Examples include planar flows [59], auto-regressive flows [36], continuous-time diffusive flow [63], neural spline flow [17] and convex potential flow [30].

One way to construct a neural network \( \mathbb{R}^d \to \mathbb{R}^d \) to approximate a flow map is to use the planar flow. A K-layer planar flow is defined by \( T = T_K \circ \ldots \circ T_1 \circ T_0 \), where \( T_k : \mathbb{R}^d \to \mathbb{R}^d \) is given by
\[
(3.29) \quad x^{k+1} = T_k(x^k) = x^k + u_k h(W_k^T x^k + b_k),
\]
where \( W_k, u_k \in \mathbb{R}^d, b_k \in \mathbb{R} \), and \( h \) is a smooth, element-wise, non-linear activation function such as tanh. Direct computation shows that
\[
(3.30) \quad J_k = \det(\nabla T_k) = 1 + h'(W_k^T x^k + b_k) u_k^T W_k.
\]
Here \( T_k \) is a diffeomorphism if \( h'(W_k^T x^k + b_k)u_k^T W_k < 1 \) for \( \forall x_k \). The determinant of the transport map can be computed as \( \det(\nabla T) = J_K J_{K-1} \ldots J_0 \). and we have \( \varphi(T(x)) = \frac{x^T(x)}{\det(\nabla T)} \). The optimization problem can be formulated in terms of \( W_k, u_k \) and \( b_k \).
Another widely used neural network architecture to approximate a flow map is the convex potential flow [30], which defines a diffeomorphism via the gradient of a strictly convex function. An Input Convex Neural Network (ICNN) [2] parameterizes an unknown convex function which guarantees the existence of diffeomorphism. A fully connected, \( K \)-layer ICNN can be parameterized by

\[
    z_{l+1} = \sigma_l(W_l z_l + A_l X + b_l), \quad l = 0, 1, \ldots K - 1,
\]

where \( z_0 = 0, \ W_0 = 0, \ \Theta = \{W_l, A_l, b_l\} \) are the parameters to be determined, and \( \sigma_l \) is the nonlinear activation function. As proved in [2], if all entries \( W_l \) are non-negative, and all \( \sigma_l \) are convex and non-decreasing, then \( f(X; \Theta) \) is a convex function with respect to \( X \). Hence \( \nabla_X f(X; \Theta) \) provides a parametric approximation to the flow map. In the current study, we adopt the idea of convex potential flow to develop the Lagrangian EVNN method. We’ll consider other types of flows in future works.

**Remark 3.5.** It is worth mentioning that the gradient of a convex function \( f \) only defines a subspace of diffeomorphism, and the optimal solution of (3.14) may not belong to this subspace.

**4. Numerical Experiments.** In this section, we test the proposed EVNN methods for various gradient flows that can be derived by the EnVarA. The numerical accuracy is evaluated through the point-wise absolute spatial errors or the absolute and relative spatial \( L^2 \)-errors between the NN solutions and the corresponding reference solutions. The analytic solutions or numerical solutions obtained by traditional numerical methods are taken as the reference solutions.

**4.1. \( L^2 \)-gradient flow.**

**4.1.1. Poisson equation.** We first consider a two-dimensional Poisson equation given by

\[
    \begin{cases}
    -\Delta u(x) = f(x), & x \in \Omega \subset \mathbb{R}^2, \\
    u(x) = 0, & x \in \partial \Omega.
    \end{cases}
\]

The corresponding variational problem can be formulated as: finding \( u \in H^1_0 \) such that

\[
    u = \arg \min_{u \in H^1_0} \int_{\Omega} \frac{1}{2} |\nabla u|^2 - f(x)u(x)dx.
\]

We can obtain an optimal solution of the variational problem by solving the following \( L^2 \)-gradient flow

\[
    \frac{d}{dt} \left( \int_{\Omega} \frac{1}{2} |\nabla u|^2 - f(x)u(x)dx + \lambda \int_{\partial \Omega} |u|^2 dS \right) = -\int_{\Omega} u_t^2 dx.
\]

Here the surface energy term \( \lambda \int_{\partial \Omega} |u|^2 dS \) is added to enforce the Dirichlet boundary condition. We take \( \lambda = 500 \) in the following numerical experiments.

We test our algorithm, Alg. 3.1 for Eq. (4.1) in the following two settings.

- **Case 1:** \( \Omega = \{(x, y) \mid x^2 + y^2 \leq 1\} \) and \( f(x) = 1 \). The exact solution is \( u(x) = 1 - 0.25(x^2 + y^2) \).
- **Case 2:** \( \Omega = (0, \pi) \times (-\pi/2, \pi/2) \) and \( f(x) = 2 \sin x \cos y \). The exact solution is \( u(x) = \sin x \cos y \).
The neural network architectures for both test cases are the same. We use a 1-block ResNet with 20 hidden nodes in hidden layers. We take $\tau = 0.01$ and regenerate samples to evaluate the numerical integration in Eq. (3.10). For Case 1, we generate sets of samples $\{x_i = (r_i \cos \theta_i, r_i \sin \theta_i)\}_{i=1}^{N}$ in polar coordinates, where $\{r_i\}_{i=1}^{N}$ and $\{\theta_i\}_{i=1}^{N}$ are drawn from the uniform distribution in $(0, 1)$ and $(0, 2\pi)$, respectively. For Case 2, we use Latin hypercube sampling (LHS) to generate samples in $\Omega$. The optimization problem at each time step is solved by a standard L-BFGS algorithm.

![Fig. 4.1: Numerical results for the two-dimensional Poisson equation. (a) - (b) The evolution of discrete free energy with respect to time. (c) - (d) The relative $L_2$-errors to the stationary solutions for the three methods.](image)

To verify the energy stability property of our algorithm, we include the plot of the free energy in each epoch in Fig. 4.1. It is clear that, in these two cases, the free energy monotonically decreases during the training process. We also compare the numerical performance of the proposed numerical method, the EVNN with the PINN and the DRM. Both PINN and DRM are trained by 50000 epochs. The corresponding relative $L_2$ errors are shown in Fig 4.1. For our EVNN, since we use at most 100 iterations in solving each optimization problem (3.10), we plot error at $t^n$ at the place of 100 epoch in Fig. 4.1. The models are evaluated at $101 \times 101$ and 7845 quadrature points, respectively. For these two problems, the relative errors computed by the EVNN are reduced to $10^{-2}$ and $10^{-1}$, respectively. Although the PINN achieves better accuracy in the constant forcing setting (Case 1), the EVNN achieves comparably better results in these two cases, especially in periodic forcing setting (Case 2).

Next, we consider a high-dimensional Poisson equation

$$-\Delta u = f(x), \quad x \in \Omega = (-1, 1)^d,$$

with homogeneous Neumann boundary condition $\frac{\partial u}{\partial n}|_{\partial \Omega} = 0$. We take $f(x) = \pi^2 \sum_{k=1}^{d} \cos(\pi x_k)$. Similar numerical examples are studied in [21] by using the deep Ritz method. The exact solution to this high-dimensional Poisson equation is $u(x) = \sum_{k=1}^{d} \cos(\pi x_k)$. Following [21], we slightly modify the free energy into the following form to fix the solution since we impose the homogeneous Neumann boundary condition in this example:

$$\mathcal{F}[u] = \int_{(-1,1)^d} \frac{1}{2} \| \nabla u \|^2 - f(u) x - \lambda \left( \int_{(-1,1)^d} f(x) \right)^2,$$
where the last term enforces $\int_{[-1,1]^d} u \, dx = 0$. We take $\lambda = 0.5$ as in [49] in all numerical tests.

Due to the large computational cost of L-BFGS in high-dimensional spaces, we solve the optimization problem at each iteration by an Adam algorithm. The model is trained with 200 outer iterations and at most 200 iterations for inner optimization (3.10) with an early stop rule specifying that the $L_2$ norm of the error of model parameters between two consecutive epochs should be less than $10^{-6}$.

| Setting | $d = 4$ | $d = 8$ | $d = 16$ | $d = 32$ |
|---------|---------|---------|---------|---------|
| $(2, 10, 1000)$ | 0.024 | 0.046 | 0.623 | 0.867 |
| $(3, 60, 1000)$ | 0.042 | 0.048 | 0.077 | 0.117 |
| $(3, 60, 10000)$ | 0.018 | 0.022 | 0.036 | 0.077 |

The numerical results with different neural network settings for Eq. (4.3) in different dimensions are summarized in Table 4.1. The entry in the first column of Table 4.1, for example $(2, 10, 1000)$, represents 2 residual blocks with 10 nodes in each of the linear layers. 1000 denotes the number of samples that we draw in each epoch. The experiment runs from dimension 4 to 32 and the corresponding $L_2$ errors are shown in the table. As we can see, our algorithm achieves comparable results in each setting. It can also be seen from the result that the width of the network plays an important role in increasing the expressive power of the neural network and the solution accuracy.

### 4.1.2. Heat equation.

Next we consider the initial-boundary value problem of a heat equation given by

\[
\begin{cases}
    u_t = \Delta u(x), \quad x \in \Omega = (0, 2)^2, \quad t \in (0, T], \\
    u(x, t) = 0, \quad x \in \partial \Omega, \quad t \in (0, T], \\
    u(x, 0) = \sin \left( \frac{\pi}{2} x_1 \right) \sin \left( \frac{\pi}{2} x_2 \right), \quad x \in \Omega = (0, 2)^2.
\end{cases}
\] (4.5)

We test the numerical accuracy of the proposed EVNN scheme over time using this problem. This heat equation problem can be viewed as an $L^2$-gradient flow satisfying the energy-dissipation law

\[
\frac{d}{dt} \left( \int_{\Omega} \frac{1}{2} |\nabla u|^2 \, dx + \lambda \int_{\partial \Omega} |u|^2 \, dS \right) = - \int |u_t|^2 \, dx.
\] (4.6)

Again, a surface energy term is added to enforce the Dirichlet boundary condition.

In the numerical simulation, we take $\tau = 0.01$ and use a 1-block ResNet with 20 nodes in each layer. The nonlinear activation function is chosen to be tanh. We also employ the L-BFGS optimizer, and the same 301 $\times$ 301 uniform mesh points as training samples in each time step in this test. The maximum number of iterations in each time step is set to be 300. To test the accuracy of the solution over time, we compare the neural network solution with the finite difference solution. In more details, the finite difference method solves $\phi_h^{n+1}$ using

\[
\phi_h^{n+1} = \arg \min_{\phi_h \in C} \frac{1}{2\tau} \|\phi_h - \phi_h^n\|^2 + \frac{1}{2} \int |\nabla h| \phi_h|^2 \, dx,
\]
where $\mathcal{C}$ is the set of grid functions defined on the computational domain. We use a $101 \times 101$ uniform grid in the FDM simulation.

We compare the NN solution with the FDM solution at the grid points of the $101 \times 101$ computational grid of the FDM method. Figure 4.2 summarizes the results of our experiment. Specifically, the top panel of Fig. 4.2 shows the NN predicted spatio-temporal solution at four different times, $t = 0.01, 0.2, 0.4$ and 0.6, respectively. The middle panel of Fig. 4.2 plots the relative error in the $L_2$ norm. Low panel of Fig. 4.2 shows the evolution of the numerical free energy with respect to time, which demonstrates the energy stability of the proposed Eulerian EVNN scheme.

**4.1.3. Allen-Cahn equation.** In this subsection, we consider to solve the Allen-Cahn equation with a volume constraint, which is extensively utilized in phase-field modeling. The phase-field modeling has become a versatile technique to solve interface problems arising from different disciplines [15]. In particular, we consider an Allen-
Fig. 4.3: Numerical results for the Allen-Cahn equation with a volume constraint. (a)-(d) NN solutions at $t = 0, 0.05, 0.1$ and $0.3$, respectively. (e)-(h) Absolute differences between NN solutions and FEM solutions at $t = 0, 0.05, 0.1$, and $0.3$, respectively. (i) Evolution of numerical free energies with respect to time for both of the FEM and EVNN solutions.

The Allen-Cahn equation

$$\begin{aligned}
\frac{\partial \varphi}{\partial t} &= - \left( \frac{F'(\varphi)}{\varepsilon^2} - \Delta \varphi(x, t) + 2W \left( \int \varphi dx - A \right) \right), \\
\varphi(x, t) &= -1, \quad x \in \partial \Omega, \quad t > 0, \\
\varphi(x, 0) &= -\tanh(10(\sqrt{4x_1^2 + x_2^2} - 0.5)), \quad x \in \Omega.
\end{aligned}$$

where $\varphi(x, t) \in [-1, 1]$ is the phase-field order parameter. $\Omega = (-1, 1)^2$. The constants $\varepsilon$, $W$ and $A$ are model parameters. The corresponding energy-dissipation law is given by

$$\frac{d}{dt} \mathcal{F}[\varphi] = - \int_{\Omega} |\varphi_t|^2 dx, \quad \mathcal{F}[\varphi] = \int_{\Omega} \frac{1}{2} |\nabla \varphi|^2 + \frac{1}{4\varepsilon^2} (\varphi^2 - 1)^2 dx + W \left( \int_{\Omega} \varphi dx - A \right)^2.$$

Here the last term is a penalty term for the volume constraint that enforces $\int \varphi dx \approx A$, as the Allen-Cahn equation does not preserve the volume fraction $\int \varphi dx$ over time. In the numerical simulation, we take $A = -(4 - \pi r^2) + \pi r^2$, $r = 0.5$, $\frac{1}{\varepsilon^2} = 100$, and $W = 1000$.

The neural network architecture used here is exactly the same as that for the heat equation. We use $201 \times 201$ uniform grid points in $(-1, 1)^2$ as training samples in the
bulk region of the domain $\Omega$. To test the numerical accuracy of the EVNN scheme for this problem, we also solve Eq. (4.7) by a finite element method, which approximates the phase-field variable $\varphi$ by a piece-wise linear function $\varphi_h(x,t) = \sum_{i=1}^{N} \gamma_i(t) \psi_i(x)$, where $\psi_i(X)$ are hat functions supported on the computational mesh. Inserting $\varphi_h$ into the continuous energy–dissipation law (4.8), we get a discrete energy–dissipation law with the discrete energy and dissipation given by

$$F_{FEM}^h = \sum_{e=1}^{N_e} \int_{\tau_e} \frac{1}{2} \left| \sum_{i=1}^{N} \gamma_i \nabla \psi_i(x) \right|^2 + \frac{1}{2} \sum_{i=1}^{N} (\gamma_i^2 - 1) \psi_i(x) \, dx,$$

and $D_{FEM}^h = \sum_{e=1}^{N_e} \int_{\tau_e} \left( \sum_{i=1}^{N} \gamma_i'(t) \psi_i(x) \right)^2 \, dx$ respectively. Here $\tau_e$ is used to denote a finite element cell, and $N_e$ is the number of cells. We also construct a piece-wise linear approximation to the nonlinear term in the discrete energy. This form of discretization was used in [71]. We can solve $\gamma_i$ at each time step by using the scheme (3.7).

The simulation results are summarized in Fig. 4.3. Numerical simulation of phase-field type models is often challenging. To capture the dynamics of the evolution of the thin diffuse interface, the mesh-size should be much smaller than $\epsilon$, the width of the diffuse interface layer. To overcome this difficulty, traditional numerical methods use the adaptive and moving mesh approaches. In contrast, neural-network-based numerical scheme has a mesh-free feature. The number of parameters of the neural network can be much smaller than the number of finite element mesh cells needed to capture the diffuse interface. Since the dimension of the optimization problem in the NN-based scheme is much smaller than that of the FEM-based scheme, the NN-based solver is much faster than the FEM-based solver which do not use adaptive or moving meshes.

4.2. Generalized diffusions. In this subsection, we apply the proposed Lagrangian EVNN scheme to solving two generalized diffusion problems: the Fokker-Planck equation and the porous medium equation, respectively.

4.2.1. Fokker-Planck equation. We first consider a Fokker-Planck equation

$$\begin{cases}
\rho_t = \nabla \cdot (\nabla \rho + \rho \nabla V), & x \in \Omega \subset \mathbb{R}^d, \quad t \in (0,T), \\
\rho(x,0) = \rho_0(x), & x \in \Omega \subset \mathbb{R}^d,
\end{cases}$$

where $V(x)$ is a prescribed potential energy function. The Fokker-Planck equation can be viewed as a generalized diffusion satisfying the energy-dissipation law

$$\frac{d}{dt} \int_{\mathbb{R}^d} \rho \ln \rho + \rho V(x) \, dx = -\int_{\mathbb{R}^d} \rho |u|^2 \, dx,$$

where the probability density $\rho$ satisfies $\rho_t + \nabla \cdot (\rho \mathbf{u}) = 0$ with $\mathbf{u}$ being the averaged velocity.

We test our numerical scheme for solving the Fokker-Planck equation in 2D and 4D spaces, respectively. For both of these numerical experiments, we adopt the NN architecture of Augmented-ICNN proposed in [30]. More specifically, we use 1 ICNN block of 6 fully connected layers with 32 hidden nodes in each layer. We choose the Gaussian-Softplus function $\sigma(x) = \sqrt{\frac{x}{2}} \left( \sqrt{2\pi} \int_{0}^{x} e^{-t^2} dt + e^{-x} + \sqrt{\pi} x \right)$ as the activation function. In addition, we choose the L-BFGS optimizer with at most 50 iterations within each time step.
For the 2D case, we use 301 × 301 grid points uniformly distributed in the domain Ω = [−3, 3]² as the initial samples. We take \( V(\mathbf{x}) \) to be
\[
V = \frac{1}{2}(\mathbf{x} - \mu_{\text{target}})^T \Sigma_{\text{target}}^{-1}(\mathbf{x} - \mu_{\text{target}}) \quad \text{with} \quad \mu_{\text{target}} = \left(\frac{1}{3}, \frac{1}{3}\right),
\]
and \( \Sigma_{\text{target}} = \begin{bmatrix}
\frac{5}{8} & -\frac{3}{8} \\
-\frac{3}{8} & \frac{5}{8}
\end{bmatrix} \). The initial condition \( \rho_0(\mathbf{x}) \) is taken as the 2D standard Gaussian \( \mathcal{N}(0, I) \). The exact solution of Eq. (4.9) takes the following analytical form
\[
(4.11) \quad \rho(\mathbf{x}, t) \sim \mathcal{N}(\mu(t), \Sigma(t)),
\]
where \( \mu(t) = (1 - e^{-4t})\mu_{\text{target}} \) and \( \Sigma(t) = \begin{bmatrix}
\frac{5}{8} + \frac{3}{8} \times e^{-8t} & -\frac{3}{8} + \frac{3}{8} \times e^{-8t} \\
-\frac{3}{8} + \frac{3}{8} \times e^{-8t} & \frac{5}{8} + \frac{3}{8} \times e^{-8t}
\end{bmatrix} \). Fig. 4.4 shows the numerical results of this 2D case. Figs. 4.4(a)-4.4(d) show the evolution of the initial samples as well as the evolution of the numerical solution over space and time. The absolute and relative \( L_2 \)-errors are shown in Figs. 4.4(e)-4.4(f).

For the 4D case, we choose the domain to be \( \Omega = [-3, 3]^4 \). We take \( V(\mathbf{x}) \) to be
\[
V = \frac{1}{2}(\mathbf{x} - \mu_{\text{target}})^T \Sigma_{\text{target}}^{-1}(\mathbf{x} - \mu_{\text{target}}) \quad \text{with} \quad \mu_{\text{target}} = \left(\frac{1}{3}, \frac{1}{3}, 0, 0\right),
\]
and \( \Sigma_{\text{target}} = \begin{bmatrix}
\frac{5}{8} & -\frac{3}{8} & 0 & 0 \\
-\frac{3}{8} & \frac{5}{8} & 0 & 0 \\
0 & 0 & \frac{4}{3} & 0 \\
0 & 0 & 0 & \frac{4}{3}
\end{bmatrix} \oplus \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \). The initial condition \( \rho_0(\mathbf{x}) \) to be the 4D standard normal distribution \( \mathcal{N}(0, I) \). The exact solution of Eq. (4.9) still follows the normal distribution with \( \mu(t) = (1 - e^{-4t})\frac{1}{3}, (1 - e^{-4t})\frac{1}{3}, 0, 0 \), and \( \Sigma(t) = \begin{bmatrix}
\frac{5}{8} + \frac{3}{8} \times e^{-8t} & -\frac{3}{8} + \frac{3}{8} \times e^{-8t} & 0 & 0 \\
-\frac{3}{8} + \frac{3}{8} \times e^{-8t} & \frac{5}{8} + \frac{3}{8} \times e^{-8t} & 0 & 0 \\
0 & 0 & \frac{4}{3} & 0 \\
0 & 0 & 0 & \frac{4}{3}
\end{bmatrix} \oplus \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix} \). Here \( \oplus \) is the direct sum. Instead of using uniform grid points in 4D to draw samples, we let the initial samples be 10000
Fig. 4.5: Numerical results for the 4D Fokker Planck equation. (a)-(d) The numerical solution in $x_1$-$x_2$ plane at $t = 0, 0.1, 0.5$ and $1$, respectively. (e)-(h) The numerical solution in $x_3$-$x_4$ plane at $t = 0, 0.1, 0.5$ and $1$, respectively. (i)-(j) The $L_2$-errors of the solution.

points randomly drawn from $\rho_0(\mathbf{x})$. Fig. 4.5 shows the evolution of samples as well the values of the numerical solution on individual samples over space and time.

Fokker-Planck type equation has a wide application in machine learning. One of the fundamental tasks in modern statistics and machine learning is to estimate or generate samples from a target distribution $\rho^*(\mathbf{x})$, which might be completely known, partially known up to the normalizing constant, empirically given by the data, or directly estimated if it is not known. Examples include Bayesian inference [7], numerical integration [50], space-filling design [56], density estimation [63], and generative learning [55]. These problems can be transformed as an optimization problem, which is to seek for a $\rho^{\text{opt}} \in \mathcal{Q}$ by solving an optimization problem

\begin{equation}
\rho^{\text{opt}} = \arg \min_{\rho \in \mathcal{Q}} D(\rho || \rho^*) ,
\end{equation}

where $\mathcal{Q}$ is the admissible set, $D(p || q)$ is a dissimilarity function that assesses the differences between two probability measure $p$ and $q$. The classical dissimilarities include the Kullback-Leibler (KL) divergence and the Maximum Mean Discrepancy (MMD). The optimal solution of the optimization problem can be obtained by solving
a Fokker-Planck type equation, given by

\begin{equation}
\rho_t = \nabla \cdot (\rho \nabla \mu), \quad \mu = \frac{\delta D(\rho||\rho^*)}{\delta \rho}.
\end{equation}

The developed numerical approach has potential applications in these machine learning problems. To illustrate this point, we consider a toy problem that widely used in the machine learning literature [30]. We take the dissimilarity function as the KL divergence

\[KL(\rho||\rho^*) = \int_\Omega \rho(x) \ln \left( \frac{\rho}{\rho^*} \right) dx,\]

then Eq. (4.13) is reduced to the Fokker-Planck equation (4.9) with

\[V(x) = -\ln \rho^*.\]

The simulation results is summarized in Fig. 4.6. It can be noticed that the proposed neural-network-based algorithm can generate weighted sample for a complicated target distribution.

**4.2.2. Porous medium equation.** Next we consider a porous medium equation (PME) \(\rho_t = \Delta \rho^\alpha\), where \(\alpha > 1\) is a constant. The PME is a typical example of nonlinear diffusion equations, which can be derived through an energy-dissipation law

\begin{equation}
\frac{d}{dt} \int_\Omega \frac{1}{\alpha - 1} \rho^\alpha dx = -\int \rho |\nabla u|^2 dx.
\end{equation}

One important feature of the PME is that the solution to the PME has a compact support at any time \(t > 0\) if the initial data has a compact support. The free boundary of the compact support moves outward with a finite speed, known as the property of finite speed propagation. As a consequence, numerical simulations of the PME are often difficult by using Eulerian methods, which may fail to capture the movement of the free boundary, and suffer from numerical oscillations [46]. In a recent work [46], the authors developed a variational Lagrangian scheme using finite element
method. In this subsection, we apply the neural-network-based Lagrangian method - the Lagrangian EVNN scheme to solve the PME.

![Numerical results for the porous medium equation.](image)

We take \( \alpha = 4 \) in the following simulation. To test the numerical accuracy of the EVNN scheme for solving the PME, we consider a 2D Barenblatt-Pattle solution to the PME. The Barenblatt-Pattle solution in \( d \)-dimensional space is given by

\[
B_\alpha(x, t) = t^{-k} \left( C_0 - \frac{k(\alpha - 1)}{2\alpha} \frac{|x|^2}{t^{2\alpha/d}} \right)^{1/(\alpha - 1)}, \quad x \in \mathbb{R}^d,
\]

where \( k = (\alpha - 1 + 2/d)^{-1}, u_+ := \max(u, 0) \), and \( C_0 \) is a constant that is related to the initial mass. This solution is radially symmetric, self-similar, and has compact support \( |x| \leq \xi_\alpha(t) \) for any finite time, where \( \xi_\alpha(t) = \sqrt{\frac{2\alpha C_0}{k(\alpha - 1)}} t^{k/d} \). We take the Barenblatt solution (4.15) with \( C_0 = 0.1 \) at \( t = 0.1 \) as the initial condition, and compare the numerical solution at \( T \) with the exact solution \( B_\alpha(x, T + 0.1) \). The set of initially drawn samples consists of two parts. One part is a set of uniform grid points inside a disk \( \{(x, y) \mid \sqrt{x^2 + y^2} < \xi_\alpha(0.1)\} \). The other part is a set of 500 points distributed with equal arc length on the free boundary. The numerical results with \( \tau = 0.005 \) are shown in Fig. 4.7. It can be noticed that the proposed neural-network-based algorithm can well approximate the Barenblatt-Pattle solution and capture the movement of free boundary.

5. Conclusion. In this paper, we develop two structure-preserving EVNN numerical schemes for simulating the \( L^2 \)-gradient flows and the generalized diffusions, respectively. These numerical schemes are directly constructed based on the prescribed continuous energy-dissipation law for the system, which describe all physics that one cares about. Neural network with time-dependent parameters is utilized to introduce a finite-dimensional approximation to the original gradient flow system. As a mesh-free approach, the neural-network-based spatial discretization may enable us to study high-dimensional gradient flows with acceptable computational costs.
Various numerical experiments are presented to demonstrate the accuracy and energy stability of the proposed numerical approaches. We will investigate different neural network architectures, sampling strategy, and apply the EVNN scheme to other complex fluid models such as Cahn-Hilliard equation and Cahn-Hilliard-Navier-Stokes equations in the future work.

Acknowledgments. C. L and Y. W were partially supported by NSF DMS-1950868 and DMS-2153029. Z. X was partially supported by the NSF CDS&E-MSS 1854779.

REFERENCES

[1] S. Adams, N. Dirr, M. Peletier, and J. Zimmer, Large deviations and gradient flows, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 371 (2013), p. 20120341.
[2] B. Amos, L. Xu, and J. Z. Kolter, Input convex neural networks, in International Conference on Machine Learning, PMLR, 2017, pp. 146–155.
[3] A. Baron, Universal approximation bounds for superposition of a sigmoid function, IEEE Transaction on Information Theory, 39 (1993), pp. 930–945.
[4] R. Bellman, Dynamic Programming, Dover Publications, 1957.
[5] J.-D. Benamou and Y. Brenier, A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem, Numerische Mathematik, 84 (2000), pp. 375–393.
[6] J.-D. Benamou, G. Carlier, and M. Laborde, An augmented lagrangian approach to wasserstein gradient flows and applications, ESAIM: Proceedings and surveys, 54 (2016), pp. 1–17.
[7] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, Variational inference: A review for statisticians, J. Am. Stat. Assoc., 112 (2017), pp. 859–877.
[8] J. Bruna, B. Pehierstorfer, and E. Vanden-Eijnden, Neural galerkin scheme with active learning for high-dimensional evolution equations, arXiv preprint arXiv:2203.01360, (2022).
[9] J. A. Carrillo, K. Craig, L. Wang, and C. Wei, Primal dual methods for wasserstein gradient flows, Foundations of Computational Mathematics, 22 (2022), pp. 389–443.
[10] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems, 2 (1989), pp. 303–314.
[11] P.-G. De Gennes and J. Prost, The physics of liquid crystals, no. 83, Oxford university press, 1993.
[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805, (2018).
[13] T. Dockhorn, A discussion on solving partial differential equations using neural networks, arXiv preprint arXiv:1904.07200, (2019).
[14] M. Doi, Onsager’s variational principle in soft matter, J. Phys.: Condens. Matter, 23 (2011), p. 284118.
[15] Q. Du and X. Feng, The phase field model for geometric moving interfaces and their numerical approximations, Handbook of Numerical Analysis, 21 (2020), pp. 425–508.
[16] Y. Du and T. A. Zaki, Evolutional deep neural network, Phys. Rev. E, 104 (2021), p. 045303.
[17] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, Neural spline flows, Advances in neural information processing systems, 32 (2019).
[18] W. E, J. Han, and A. Jentzen, Algorithms for solving high dimensional pdes: From nonlinear Monte Carlo to machine learning, Nonlinearity, 35 (2021), p. 278.
[19] W. E, C. Ma, and L. Wu, Machine learning from a continuous viewpoint, i, Science China Mathematics, 63 (2020), pp. 2233—2266.
[20] W. E, C. Ma, L. Wu, and S. Wojtowytsch, Towards a mathematical understanding of neural network-based machine learning: What we know and what we don’t, CSIAM Transactions on Applied Mathematics, 1 (2020), pp. 561–615.
[21] W. E and B. Yu, The deep ritz method: A deep learning-based numerical algorithm for solving variational problems, Communications in Mathematics and Statistics, 6 (2018).
[22] B. Eisenberg, Y. Hyon, and C. Liu, Energy variational analysis of ions in water and channels: Field theory for primitive models of complex ionic fluids, The Journal of Chemical Physics, 133 (2010), p. 104104.
[23] M.-H. Giga, A. Kirshtein, and C. Liu, Variational modeling and complex fluids, Handbook
of mathematical analysis in mechanics of viscous fluids, (2017), pp. 1–41.

[24] E. D. Giorgi, Movimenti minimizzanti, in Proceedings of the Conference on Aspetti e problemi della Matematica oggi, Lecce, 1992.

[25] J. Han, A. Jentzen, and W. E, Solving high-dimensional partial differential equations using deep learning, Proceedings of the National Academy of Sciences, 115 (2018), pp. 8505–8510.

[26] B. Hanin, Which neural architectures give rise to exploding and vanishing gradients?, in NeurIPS, 2018.

[27] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[28] ———, Identity mappings in deep residual networks, in European conference on computer vision, Springer, 2016, pp. 630–645.

[29] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal processing magazine, 29 (2012), pp. 82–97.

[30] C.-W. Huang, R. T. Chen, C. Tsirigotis, and A. Courville, Convex potential flows: Universal probability distributions with optimal transport and convex optimization, arXiv preprint arXiv:2012.05942, (2020).

[31] H. J. Hwang, C. Kim, M. S. Park, and H. Son, The deep minimizing movement scheme, arXiv preprint arXiv:2109.14851, (2021).

[32] K. Jiang and P. Zhang, Numerical methods for quasicrystals, J. Comput. Phys., 256 (2014), pp. 428–440.

[33] R. Jordan, D. Kinderlehrer, and F. Otto, The variational formulation of the Fokker–Planck equation, SIAM J. Math. Anal., 29 (1998), pp. 1–17.

[34] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, hp-vpinns: Variational physics-informed neural networks with domain decomposition, Computer Methods in Applied Mechanics and Engineering, 374 (2021), p. 113547.

[35] Y. Khoo, J. Lu, and L. Ying, Solving parametric PDE problems with artificial neural networks, European Journal of Applied Mathematics, 32 (2021), pp. 421–435.

[36] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, Improved variational inference with inverse autoregressive flow, Advances in neural information processing systems, 29 (2016).

[37] J. F. Kolen and S. C. Kremer, Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies, 2001, pp. 237–243.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in Advances in neural information processing systems, 2012, pp. 1097–1105.

[39] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE transactions on neural networks, 9 (1998), pp. 987–1000.

[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[41] R. Lifshitz and H. Diamant, Soft quasicrystals–why are they stable?, Philosophical Magazine, 87 (2007), pp. 3021–3030.

[42] F. H. Lin and C. Liu, Static and dynamic theories of liquid crystals, J. Partial Differential Equations, 14 (2001), pp. 289–330.

[43] S. Lisini, D. Matthes, and G. Savaré, Cahn–Hilliard and thin film equations with nonlinear mobility as gradient flows in weighted-wasserstein metrics, Journal of differential equations, 253 (2012), pp. 814–850.

[44] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, A survey on deep learning in medical image analysis, Medical image analysis, 42 (2017), pp. 60–88.

[45] C. Liu and H. Sun, On energetic variational approaches in modeling the nematic liquid crystal flows, Discrete & Continuous Dynamical Systems, 23 (2009), p. 455.

[46] C. Liu and Y. Wang, On lagrangian schemes for porous medium type generalized diffusion equations: a discrete energetic variational approach, J. Comput. Phys., 417 (2020), p. 109566.

[47] ———, A variational lagrangian scheme for a phase-field model: A discrete energetic variational approach, SIAM J. Sci. Comput., 42 (2020), pp. B1541–B1569.

[48] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, DeePDE: A deep learning library for solving differential equations, SIAM Review, 63 (2021), pp. 208–228.
[49] Y. Lu, J. Lu, and M. Wang, A priori generalization analysis of the deep ritz method for solving high dimensional elliptic partial differential equations, in Conference on Learning Theory, PMLR, 2021, pp. 3196–3241.

[50] T. Müller, B. McWilliams, F. Rousseau, M. Gross, and J. Novák, Neural importance sampling, ACM Transactions on Graphics (TOG), 38 (2019), pp. 1–19.

[51] J. Noh, Y. Wang, H.-L. Liang, V. S. R. Jampani, A. Majumdar, and J. P. Lagerwall, Dynamic tuning of the director field in liquid crystal shells using block copolymers, Phys. Rev. Res., 2 (2020), p. 033160.

[52] T. Ohta and K. Kawasaki, Equilibrium morphology of block copolymer melts, Macromolecules, 19 (1986), pp. 2621–2632.

[53] L. Onsager, Reciprocal relations in irreversible processes. I., Phys. Rev., 37 (1931), p. 405.

[54] ———, Reciprocal relations in irreversible processes. II., Phys. Rev., 38 (1931), p. 2265.

[55] G. Papamakarios, E. Talvinsick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference., J. Mach. Learn. Res., 22 (2021), pp. 1–64.

[56] L. Pronzato and W. G. Müller, Design of computer experiments: space filling and beyond, Statistics and Computing, 22 (2012), pp. 681–701.

[57] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, arXiv preprint arXiv:1711.10561, (2017).

[58] L. Rayleigh, Some general theorems relating to vibrations, Proceedings of the London Mathematical Society, 4 (1873), pp. 357–368.

[59] D. Rezende and S. Mohamed, Variational inference with normalizing flows, in International conference on machine learning, PMLR, 2015, pp. 1530–1538.

[60] G. M. Rotskoff, A. R. Mitchell, and E. Vanden-Eijnden, Active importance sampling for variational objectives dominated by rare events: Consequences for optimization and generalization, arXiv preprint arXiv:2008.06334, (2020).

[61] Z. Shen, H. Yang, and S. Zhang, Nonlinear approximation via compositions, arXiv preprint arXiv:1902.10170, (2019).

[62] J. Sirignano and K. Spiliopoulos, Dgm: A deep learning algorithm for solving partial differential equations, J. Comput. Phys., 375 (2018), pp. 1339–1364.

[63] E. G. Tabak, E. Vanden-Eijnden, et al., Density estimation by dual ascent of the log-likelihood, Communications in Mathematical Sciences, 8 (2010), pp. 217–233.

[64] B. P. van Milligen, V. Tribaldos, and J. Jiménez, Field theory of reaction-diffusion: Law of mass action with an energetic variational approach, Phys. Rev. E, 102 (2020), p. 062147.

[65] Y. Wang, T.-F. Zhang, and C. Liu, Onsager’s variational principle in active soft matter, Soft Matter, 17 (2021), pp. 3634–3653.

[66] Y. Wang, J. Chen, C. Liu, and L. Kang, Particle-based energetic variational inference, Stat. Comput., 31 (2021), pp. 1–17.

[67] Y. Wang and C. Liu, Some recent advances in energetic variational approaches, Entropy, 24 (2022).

[68] Y. Wang, C. Liu, P. Liu, and B. Eisenberg, Field theory of reaction-diffusion: Law of mass action with an energetic variational approach, Phys. Rev. E, 102 (2020), pp. 062147.

[69] Y. Wang, T.-F. Zhang, and C. Liu, A two species micro-macro model of wormlike micellar solutions and its maximum entropy closure approximations: An energetic variational approach, Journal of Non-Newtonian Fluid Mechanics, 293 (2021), p. 104559.

[70] Q. Wei, Y. Jiang, and J. Z. Y. Chen, Machine-learning solver for modified diffusion equations, Phys. Rev. E, 98 (2018), p. 053304.

[71] J. Xu, Y. Li, S. Wu, and A. Bousquet, On the stability and accuracy of partially and fully implicit schemes for phase field modeling, Computer Methods in Applied Mechanics and Engineering, 345 (2019), pp. 826–853.

[72] S. Xu, P. Sheng, and C. Liu, An energetic variational approach for ion transport, Communications in Mathematical Sciences, 12 (2014), pp. 779–789.

[73] S. Xu, Z. Xu, O. V. Kim, R. I. Litvinov, J. W. Weisel, and M. S. Alber, Model predictions of deformation, emulsification and permeability of partially obstructive blood clots under variable shear flow, Journal of The Royal Society Interface, 14 (2017).

[74] X. Xu, Y. Di, and M. Doi, Variational method for liquids moving on a substrate, Physics of Fluids, 28 (2016), p. 081701.

[75] P. Yue, J. J. Feng, C. Liu, and J. Shen, A diffuse-interface method for simulating two-phase flows of complex fluids, J. Fluid Mech., 515 (2004), pp. 293–317.

[76] Y. Zang, G. Bao, X. Ye, and H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, J. Comput. Phys., 411 (2020), p. 109409.