Neural Conservation Laws: A Divergence-Free Perspective

Anonymous Author(s)
Affiliation
email

Abstract

We investigate the significance and parameterization of deep neural networks that are by design divergence-free. To motivate the importance of divergence-free models, we observe that the continuity equation, a fundamental partial differential equation in the description of conservation laws, can be characterized through a divergence-free vector field. We hence propose an approach to building divergence-free neural networks through the concept of differential forms, and with the aid of automatic differentiation, realize two practical constructions with differing trade offs. Furthermore, we show these models are universal and can be used to represent any divergence-free vector field if given sufficient capacity. As a result, we can parameterize density and vector fields that always satisfy the continuity equation simply by design, foregoing the need for expensive numerical simulation. We experimentally validate our approaches on neural network-based solutions to fluid equations and on learning dynamical Optimal Transport maps.

1 Introduction

Modern successes in deep learning are often attributed to the expressiveness of black-box neural networks. These models are known to be universal function approximators [Hornik et al., 1989]. This flexibility, however, comes at a cost. In contrast to other parametric function approximators such as Finite Elements [Schroeder and Lube, 2017], it is hard to design networks that have *baked-in* constraints at a structural level. Existing approaches often resort to penalty methods for constraints—but these increase the cost of training and can produce inaccuracies in downstream applications when the constraints are not perfectly satisfied. Theoretical analysis of soft-constrained models also becomes more difficult, as constraints cannot be assumed to be exactly satisfied. On the other hand, enforcing hard constraints on the architecture can be difficult, and even once enforced, it is often unclear whether the model remains sufficiently expressive within the constrained function class.

In this work, we discuss a method to directly bake in two constraints into deep neural networks: having a divergence of zero, and satisfying the continuity equation. The former directly leads into the latter, so the first portion of the paper focuses on divergence-free vector fields. These represent a special class of vector fields which have widespread use in the physical sciences, *e.g.*, Maxwell's equations from electromagnetism describe the magnetic field as divergence-free, and in fluid dynamics, divergence-free vector fields are used to model incompressible fluid interactions formalized by the Euler or Navier-Stokes equations.

While potential fields—complements of divergence-free fields (in a sense made precise by the Hodge decomposition)—can be parameterized by using automatic differentiation to compute the gradient of a scalar parametric function, a general construction for divergence free fields is missing from the machine learning literature. However, in $\mathbb{R}^3$ we know that the curl of a vector field has a divergence of zero, which has seen many uses in graphics simulations [Eisenberger et al., 2018]. Behind this fact is the powerful identity $d^2 = 0$, a generalization of divergence and curl

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

through differential forms. We first explore this generalization, then discuss a construction derived from it for parmaterizing divergence-free vector fields. While this construction has been partially discussed previously [Barbarosie, 2011, Kelliher, 2021], it is not extensively known and has not been practically validated in the machine learning community. Concretely, we derive an approach to transform sufficiently smooth neural networks into divergence-free neural networks. This can be efficiently done with automatic differentiation and recent advances in practical composable transformations [Bradbury et al., 2018]. Furthermore, we show that our approaches can theoretically represent any divergence-free vector field.

We combine this new modeling tool with a novel observation that solutions of the continuity equation, a partial differential equation describing the relation between a density and a flow that transforms the density, can be characterized jointly as a divergence-free vector field. As a result, we can parameterize neural networks that, by design, always satisfy the continuity equation, *i.e.*, Neural Conservation Laws (NCL). This allows us to forego extra penalty terms and expensive numerical simulations. We validate our approaches on modeling fluid equations and learning dynamical optimal transport maps.

2 Preliminaries: Differential forms in $\mathbb{R}^n$

We are interested in learning divergence free vector fields in $\mathbb{R}^n$. We will use the notation of *differential forms* in $\mathbb{R}^n$ that will help derivations and proofs in the paper. Below we provide basic definitions and properties of differential forms, for more extensive introduction see *e.g.*, [Do Carmo, 1998, Morita, 2001].

We let $x = (x_1, \dots, x_n) \in \mathbb{R}^n$, and $dx_1, \dots, dx_n$ the coordinate differentials, *i.e.*, $dx_i(x) = x_i$ for all $i \in [n] = \{1, \dots, n\}$. The linear vector space of k-forms in $\mathbb{R}^n$, denoted $\Lambda^k(\mathbb{R}^n)$, is the space of k-linear alternating maps

$$\varphi : \overbrace{\mathbb{R}^n \times \cdots \times \mathbb{R}^n}^{k \text{ times}} \to \mathbb{R} \tag{1}$$

A k-linear alternating map φ is linear in each of its coordinates and satisfies $\varphi(\dots, v, \dots, u, \dots) = -\varphi(\dots, u, \dots, v, \dots)$. The space $\Lambda^k(\mathbb{R}^n)$ is a linear vector space with a basis of alternating k-forms denoted $dx_{i_1} \wedge \cdots \wedge dx_{i_k}$. The way these k-forms act on k-vectors, $v_1, \dots, v_k \in \mathbb{R}^n$ is as a signed volume function:

$$dx_{i_1} \wedge \cdots \wedge dx_{i_k}(v_1, \dots, v_k) = \det \left[dx_{i_r}(v_s)\right]_{r,s \in [k]} \tag{2}$$

Expanding an arbitrary element $\omega \in \Lambda^k(\mathbb{R}^n)$ in this basis gives

$$\omega = \sum_{i_1 < i_2 < \cdots < i_k} a_{i_1 \cdots i_k}\, dx_{i_1} \wedge \cdots \wedge dx_{i_k} = \sum_I a_I\, dx_I \tag{3}$$

where $i_1, \dots, i_k \in [n]$, $I = (i_1, \dots, i_k)$ are multi-indices, a_I are scalars, and $dx_I = dx_{i_1} \wedge \cdots \wedge dx_{i_k}$.

The space of *differential* k-forms (also called k-forms in short), denoted $\mathcal{A}^k(\mathbb{R}^n)$, is defined by smoothly assigning to each $x \in \mathbb{R}^n$ a k-linear alternating form $w \in \Lambda^k(\mathbb{R}^n)$. That is

$$w(x) = \sum_I f_I(x) dx_I \tag{4}$$

where $f_I : \mathbb{R}^n \to \mathbb{R}$ are smooth scalar functions. Note that $\mathcal{A}^0(\mathbb{R}^n)$ is the space of smooth scalar functions over $\mathbb{R}^n$. The differential operator can be seen as a linear operator $d : \mathcal{A}^0(\mathbb{R}^n) \to \mathcal{A}^1(\mathbb{R}^n)$ defined by

$$df(x) = \sum_{i=1}^n \frac{\partial f}{\partial x_i}(x) dx_i \tag{5}$$

The exterior derivative $d : \mathcal{A}^k(\mathbb{R}^n) \to \mathcal{A}^{k+1}(\mathbb{R}^n)$ is a linear differential operator generalizing the differential to arbitrary differential k-forms:

$$d\omega(x) = \sum_I df_I \wedge dx_I \tag{6}$$

where the exterior product $\omega \wedge \eta$ of two forms $\omega = \sum_I f_I dx_I$, $\eta = \sum_J g_J dx_J$ is defined by extending equation 2 linearly, that is, $\omega \wedge \eta = \sum_{I,J} f_I g_J dx_I \wedge dx_J$. An imporant property of the

exterior derivative is that $dd\omega = 0$ for all ω. This property can be checked using the definition in equation 6 and the symmetry of mixed partials, $\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$.

The hodge operator $\star : \mathcal{A}^k(\mathbb{R}^n) \to \mathcal{A}^{n-k}(\mathbb{R}^n)$ matches to each k-form an $n-k$-form by extending the rule

$$\star\,(dx_I) = (-1)^\sigma dx_J \tag{7}$$

linearly, where $\sigma = \mathrm{sign}([I, J])$ is the sign of the permutation $(I, J) = (i_1, \ldots, i_k, j_1, \ldots, i_{n-k})$ of $[n]$. The hodge star is (up to a sign) its own inverse, $\star\star\omega = (-1)^{k(n-k)}\omega$, for all $\omega \in \mathcal{A}^k(\mathbb{R}^n)$. The d operator has an adjoint operator $\delta : \mathcal{A}^k(\mathbb{R}^n) \to \mathcal{A}^{k-1}(\mathbb{R}^n)$ defined by $\delta = (-1)^{n(k+1)+1} \star d\star$.

A vector field $v(x) = (v_1(x), \ldots, v_n(x))$ in $\mathbb{R}^n$ can be identified with a 1-form $v = \sum_{i=1}^n v_i(x)dx_i$. The divergence of v, denoted $\mathrm{div}(v)$ can be expressed using exterior derivative :

$$d \star v = d \sum_{i=1}^n v_i \star dx_i = \sum_{i=1}^n \frac{\partial v_i}{\partial x_i} dx_1 \wedge \cdots \wedge dx_n = \mathrm{div}(v) dx_1 \wedge \cdots \wedge dx_n, \tag{8}$$

where in the second equality we used the definition of $\star$, the definition of d and the fact that $dx_i \wedge dx_i = 0$ (can be seen from the repeating rows in the matrix inside the determinant in equation 2). Note that the n-form $\mathrm{div}(v)dx_1 \wedge \cdots \wedge dx_n$ is identified via $\star$ with the function (*i.e.*, 0-form) $\mathrm{div}(v)$.

3 Parameterizing Divergence Free Vector Fields

To parameterize a divergence free vector field $v(x) = (v_1(x), \ldots, v_n(x))$, we note that from the properties of differential forms introduced above, taking an arbitrary (n-2)-form $\mu \in \mathcal{A}^{n-2}(\mathbb{R}^n)$ we have that

$$0 = dd\mu = d(d\mu)$$

and therefore according to equation 8 the vector field (1-form) v defined via $\star v = d\mu$ is divergence free. Applying the hodge star to both sides we get that $v = (-1)^{n-1} \star d\mu$ is divergence free and the same is true without the constant $(-1)^{n-1}$, *i.e.*,

$$v = \star d\mu \tag{9}$$

Lets write the parameterization $v = \star d\mu$ explicitly. First $\mu \in \mathcal{A}^{n-2}(\mathbb{R}^n)$. Since a basis for $\mathcal{A}^{n-2}(\mathbb{R}^n)$ can be chosen to be $\star(dx_i \wedge dx_j)$ for all $i < j$ we can write $mu = \sum_{i<j} \mu_{ij} \star (dx_i \wedge dx_j)$. Now a direct calculation shows that up to a global sign

$$\star d\mu = \sum_{i=1}^n \left[\sum_{j>i} \frac{\partial \mu_{ij}}{\partial x_j} - \sum_{j<i} \frac{\partial \mu_{ji}}{\partial x_j} \right] dx_i$$

This formula leads to a simple matrix formulation: If we let $A : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ be the anti-symmetric matrix-valued function where $A_{ij} = \mu_{ij}$ for $i > j$ and $A_{ij} = -\mu_{ij}$ then the divergence free vector field $v = \star d\mu$ can be written as taking row-wise divergence of A, *i.e.*,

$$v = \begin{pmatrix} \mathrm{div}(A_1) \\ \vdots \\ \mathrm{div}(A_n) \end{pmatrix} \tag{10}$$

The $\mathrm{div}(v) = 0$ can also be check directly from equation 10.

A more efficient, yet equally expressive model, for divergence free vector fields f can be derived using only n function. This is in contrast to the $O(n^2)$ functions required in equation 10. The idea of this construction is to model $\nu \in \mathcal{A}^{n-2}(\mathbb{R}^n)$ as $\nu = \delta\nu$, where $\nu \in \mathcal{A}^{n-1}(\mathbb{R}^n)$. Putting this together with equation 9 we get that

$$v = \star d\delta\nu \tag{11}$$

is a divergence free vector field. To provide equation 11 in matrix formulation we first write $\nu \in \mathcal{A}^{n-1}(\mathbb{R}^n)$ in the n-1-form basis, *i.e.*, $\nu = \sum_{i=1}^n \nu_i \star dx_i$. Then a direct calculation provides up to a global sign

$$\delta\nu = \sum_{i<j} \left[\frac{\partial \nu_i}{\partial x_j} - \frac{\partial \nu_j}{\partial x_i} \right] \star (dx_i \wedge dx_j)$$

This means that A takes the form

$$A = Jb - Jb^T \tag{12}$$

where $b = (b_1, \ldots, b_n) : \mathbb{R}^n \to \mathbb{R}^n$ is a vector-valued function, and $Jb = [\frac{\partial b_i}{\partial x_j}]_{i,j}$ is its matrix of partial derivatives. By using automatic differentiation and recent advances in composable function transforms, we can compute this expression and then its divergence.

To summarize, we have two constructions for divergence free vector fields v:

Matrix-field: (equations 9 and 10) v is represented using an anti-symmetric matrix field A.

Vector-field: (equations 11 and 10+12) v is represented using a vector field b.

As we show in the next section, these two approaches are maximally expressive (universal), meaning they can approximate arbitrary smooth divergence free vector fields. While theoretically equivalent, empirically these two constructions can exhibit different practical tradeoffs. The implicit construction has a clear computational advantage due to the compactness of its representation— $O(n)$ vs. $O(n^2)$ components to represent—the explicit construction requires one less derivative computation. This can also have practical implications once we discuss the use of neural networks for representing μ or ν.

3.1 Universality

Both matrix-field and vector-field divergence free representations are universal, namely, can model arbitrary divergence free vector fields. The main tool in the proof is the Hodge decomposition theorem [Morita, 2001, Berger, 2003] . For simplicity we will be working on a periodic piece of $\mathbb{R}^n$, namely the torus $\mathbb{T} = [-M, M]^n / \sim$ where $\sim$ means identifying opposite edges of the n-dimensional cube, and $M > 0$ is arbitrary. Vector fields with compact support can always be fitted in $\mathbb{T}$ with $M > 0$ sufficiently large. Since $\mathbb{T}$ is locally identical to $\mathbb{R}^n$, all the previous definitions and constructions hold.

Theorem 3.1. *The matrix and vector-field representations are universal in $\mathbb{T}$, possibly only missing a constant vector field.*

Proof. First, consider a divergence free vector field and its representation as a 1-form $v \in \mathcal{A}^1(\mathbb{T})$. Then v is divergence free means $d \star v = 0$. We denote by $c = (c_1, \ldots, c_n) \in \mathbb{R}^n$ a constant vector field; note that c is also a well defined vector field over $\mathbb{T}$. We will use the notation c to denote the corresponding constant 1-form. We claim $\star v = d\mu + \star c$, where $\mu \in \mathcal{A}^{n-2}(\mathbb{T})$, and $\star c \in \mathcal{A}^{n-1}(\mathcal{S})$ is constant. This can be shown with Hodge decomposition [Morita, 2001] of $\star v \in \mathcal{A}^{n-1}(\mathbb{T})$:

$$\star v = d\mu + \delta\tau + h,$$

where $\mu \in \mathcal{A}^{n-2}(\mathbb{T})$, $\tau \in \mathcal{A}^n(\mathbb{T})$ and $h \in \mathcal{A}^{n-1}$ is a harmonic n-1-form. Note that the harmonic n-1-forms over $\mathcal{T}$ are simply constant. Taking d of both sides leads to

$$d \star v = d\delta\tau.$$

Since we assumed v is div-free, $d \star v = 0$ and we get $d\delta\tau = 0$. Since $d\delta\tau = 0 = \delta\delta\tau$ then $\delta\tau$ is harmonic (constant) as well.

So far we showed $\star v = d\mu + \star c$, which shows the universality of the matrix-field representation (up to a constant). To show universality of the vector-field representation we need to show that:

$$\left\{ d\mu + \star c \;\middle|\; \mu \in \mathcal{A}^{n-2}(\mathbb{T}) \right\} = \left\{ d\delta\nu + \star c \;\middle|\; \nu \in \mathcal{A}^{n-1}(\mathbb{T}) \right\}$$

The left inclusion $\supset$ is true since $\delta\nu \in \mathcal{A}^{n-2}(\mathcal{T})$. For the right inclusion $\subset$ we take an arbitrary $\mu \in \mathcal{A}^{n-2}(\mathbb{T})$ and decompose it with Hodge: $\mu = d\omega + \delta\nu + h$, where $\omega \in \mathcal{A}^{n-3}(\mathbb{T})$, $\nu \in \mathcal{A}^{n-1}(\mathbb{T})$, and $h \in \mathcal{A}^{n-2}(\mathbb{T})$ is harmonic. Taking d of both sides leaves us with $d\mu = d\delta\nu$ that shows that $d\mu + \star c$ is included in the right set.

□

3.2 Neural Conservation Laws

We next discuss a key aspect of our work, on parameterizing solutions of conservation laws, as an application for divergence-free vector fields. Conservation laws are a focal point of mathematical

physics and have seen its uses in machine learning, with the most well known examples being scalar quantities often referred to as density, energy, or mass. Formally, a conservation law can be expressed as a first-order PDE which can be written in the form

$$\frac{\partial \rho}{\partial t} + \text{div}(f(\rho)) = 0 \tag{13}$$

where $f(\rho)$ is known as the flux, and the div is taken over spatial variables. The scalar continuity equation consists of taking $f(\rho) = \rho u$ for some given vector field u. One can then interpret the equation to mean that u is a flow which transports $\rho(0, \cdot)$ to $\rho(t, \cdot)$. Such an interpretation plays a key role in physics simulations as well as the dynamic formulation of optimal transportation [Benamou and Brenier, 2000].

We note that this equation can be expressed as a divergence-free field by augmenting the spatial dimensions with the time variable, resulting in a vector field v that maps from (t, x) to $(\rho, f(\rho))$. That is, equation 13 is equivalent to the equation

$$\text{div}(v) = \text{div}\begin{pmatrix} \rho \\ f(\rho) \end{pmatrix} = 0 \tag{14}$$

where the divergence operator is now taken with respect to (t, x), *i.e.* $\frac{\partial}{\partial t} + \sum_{i=1}^{n} \frac{\partial}{\partial x_i}$. We thus propose modeling solutions of conservation laws by parameterizing the divergence-free vector field v. This allows us to enforce the continuity equation at an architecture level. Specifically in the case of the continuity equation, we parameterize a divergence-free vector field v and set $v_1 = \rho$ and $v_{2:n+1} = \rho u$, allowing us to recover the vector field as $u = \frac{v_{2:n+1}}{\rho}$. This allows us to forego computationally expensive procedures for approximating or simulating ρ given the flow.

4 Related Works

Deep learning Existing approaches to enforcing constraints in deep neural networks can induce constraints on the derivatives, such as convexity [Amos et al., 2017] or Lipschitz continuity [Miyato et al., 2018]. More complicated formulations involve solving numerical problems such as using solutions of convex optimization problems [Amos and Kolter, 2017], solutions of fixed-points iterations [Bai et al., 2019], and solutions of ordinary differential equations [Chen et al., 2018]. These models can help provide more efficient approaches or alternative algorithms for handling downstream applications such as constructing flexible density models [Chen et al., 2019, Lu et al., 2021] or approximating dynamical optimal transport paths [Tong et al., 2020, Onken et al., 2021]. However, in many cases, there is a need to solve a numerical problem in which the solution may only be approximated up to some numerical accuracy; for instance, the need to compute the density flowing through a vector field under the continuity equation for generative modeling [Chen et al., 2018].

Applications of differential forms Differential forms and more generally, Differential Geometry, have been previously applied in manifold learning – see [Arvanitidis et al., 2021] and [Bronstein et al., 2017] for an in-depth overview. Most of the applications thus far have been restricted to 2 or 3 dimensions – either using identities like $\text{div} \circ \text{curl} = 0$ in 3D for fluid simulations [Rao et al., 2020], or for learning geometric invariances in 2d images or 3-space [Gerken et al., 2021, Li et al., 2021].

5 Conservation Laws for the Variable Density Euler Equations

As a demonstration of our method, we apply it to neural-based PDE simulations of fluid dynamics. First, we apply it to a problem consisting of modelling idealized fluid flow on the flat Torus, then, to showcase the ability to handle boundaries, another in the open unit ball $\mathbb{B} \subseteq \mathbb{R}^3$. While these are toy examples, they demonstrate the value of our method in comparison to existing approaches—namely that we can exactly satisfy the continuity equation and preserve exact mass.

The Euler equations of incompressible flow The incompressible Euler equations [Feynman et al., 1989] form an idealized model of inviscid fluid flow, governed by the system of partial differential

equations[1]

$$\frac{\partial \rho}{\partial t} + \text{div}(\rho u) = 0, \qquad \frac{\partial u}{\partial t} + \nabla_u u = \frac{\nabla p}{\rho}, \qquad \text{div}(u) = 0 \tag{15}$$

in three unknowns: the fluid velocity $u(t,x) \in \mathbb{R}^3$, the pressure $p(t,x)$, and the fluid density $\rho(t,x)$. While the fluid velocity and density are usually given at $t = 0$, the initial pressure is not required. Typically, on a bounded domain $\Omega \subseteq$, these are supplemented by the *free-slip* boundary condition and initial conditions

$$u \cdot n = 0 \text{ on } \partial\Omega \qquad u(0,x) = u_0 \text{ and } \rho(0,x) = \rho_0 \qquad \text{on } \Omega \tag{16}$$

The density ρ plays a critical role since in addition to being a conserved quantity, it influences the dynamics of the fluid evolution over time. In numerical simulations, satisfying the continuity equation as closely as possible is desirable since equations in equation 15 are coupled. Error in the density feeds into error in the velocity and then back into the density over time. In the finite element literature, a great deal of effort has been put towards developing conservative schemes that preserve mass (or energy in the more general compressible case)—see Guermond and Quartapelle [2000] and Almgren et al. [1998] for an overview. But since the application of physics informed neural networks (PINNs) to fluid problems is much newer, conservative constraints have only been incorporated as penalty terms into the loss [Mao et al., 2020, Jin et al., 2021].

5.1 Physics Informed Neural Networks

Physics Informed Neural Networks (PINNs; Raissi et al. [2019, 2017]) have recently received renewed attention as an application of deep neural networks. While using neural networks as approximate solutions to PDE had been explored before Lagaris et al. [1998], modern advances in automatic differentiation algorithms have made the application to much more complex problems feasible [Raissi et al., 2019]. The "physics" in the name is derived from the incorporation of *physical* terms into the loss function, which consist of adding the squared residual norm of a PDE. For example, to train a neural network $\phi = [\rho, p, u]$ to satisfy the Euler equations, the standard choice of loss to fit to is

$$\begin{aligned} L_F &= \left\| u_t + \nabla_u u + \frac{\nabla p}{\rho} \right\|_\Omega^2 \quad L_{\text{div}} = \|\text{div}(u)\|_\Omega \quad L_I = \|u(0,\cdot) - u_0(0,\cdot)\|_\Omega^2 + \|\rho(0,\cdot) - \rho_0(0,\cdot)\|_\Omega \\ L_{\text{Cont}} &= \left\| \frac{\partial \rho}{\partial t} + \text{div}(\rho u) \right\|_\Omega^2 \qquad L_G = \|u \cdot n\|_{\partial\Omega}^2 \qquad L_{\text{total}} = \gamma \cdot [L_F, L_I, L_{\text{div}}, L_{\text{Cont}}, L_G] \end{aligned}$$

where $\gamma = (\gamma_F, \gamma_I, \gamma_{\text{div}} \gamma_{Cont}, \gamma_G)$ denotes suitable coefficients (hyperparameter). The loss term L_G ensures fluid does not pass through boundaries, when they are present. Similar approaches were taken in [Mao et al., 2020] and [Jagtap et al., 2020] for related equations. While schemes of this nature are very easy to implement, they have the drawback that since PDE terms are only penalized and not strictly enforced, one cannot make guarantees as to the properties of the solution.

To showcase the ability of our method to model conservation laws, we will parameterize the density and vector field as $v = [\rho, \rho u]$, as detailed in Section 3.2. This means we can omit the term L_{Cont} as described in Section-5.1 from the training loss. The divergence penalty, L_{div} remains when modeling incompressible fluids, since u is not necessarily itself divergence-free. In order to stablize training, we can modify the loss terms L_F, L_G, L_I to avoid division by ρ. This is detailed in Appendix B.1.

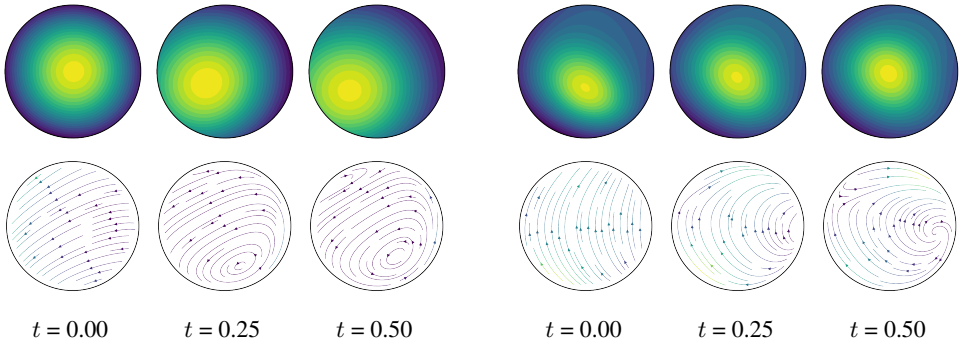
5.2 Incompressible Variable-Density inside the 3D Unit Ball

We first construct a simple unit within a bounded domain, specifically, we will consider the Euler equations inside $B(0,1) \subseteq \mathbb{R}^3$, with the initial conditions

$$\rho(0,x) = 3/2 - \|x\| \qquad v(0,x) = (-2, x_1 - 1, 1/2) \tag{17}$$

As a baseline, we compare against a parameterization that uses the divergence-free vector field only for the incompressible constraint. This is equivalent to the curl approach described in Rao et al. [2020]; another neural network is used to parameterize ρ. In contrast, our generalized formulation of divergence-free fields allow us to model the continuity equation is this case as it requires four dimensional inputs. We parameterize our method using $v = [\rho, \rho u]$, as detailed in 3.2, with the vector-field approach 3.

[1] The convective derivative appearing in equation 15, $\nabla_u u(x) = \lim_{h \to 0} \frac{u(x + hu(x)) - u(x)}{h} = [Du](u)$ is also often written as $(\nabla \cdot u)u$.



(a) NCL (Ours) (b) Divergence-free only baseline

Figure 1: 2D slices ($Z = 0$) of evolution of solution to Euler equation with initial conditions in 5.2. Compared here is our Neural Conservation Laws formulation which exactly satisfies the continuity equation (*left*), and a curl formulation that satisfies the divergence-free constraint (*right*). Both are fit to the system as described in 5.2, but the curl formulation fails to model the flow of the density under the velocity.

The results for training both networks in the described manner are shown in Figure 1. Surprisingly, even though both architectures achieve similar training loss after a comparable number of iterations (see Appendix C), the curl approach fails to advect the density accurately, whereas our conservation approach evolves the density as it should given the initial conditions.

5.3 Incompressible Variable Density Flow on $\mathbb{T}^2$

As a more involved experiment, we model the Euler equations on the flat 2-Torus $\mathbb{T}^2$. This was parameterized as a multilayer perceptron composed with the periodic embedding $\iota : [0,1]^2 \to \mathbb{T}^2$

$$\iota(x,y) = [\theta(x,y,1) \cdots \theta(x,y,N)]$$
$$\text{where: } \theta(x,y,k) = \left[\cos(2^{k+1}\pi x), \sin(2^{k+1}\pi x), \cos(2^{k+1}\pi y), \sin(2^{k+1}\pi y)\right]/2^k$$

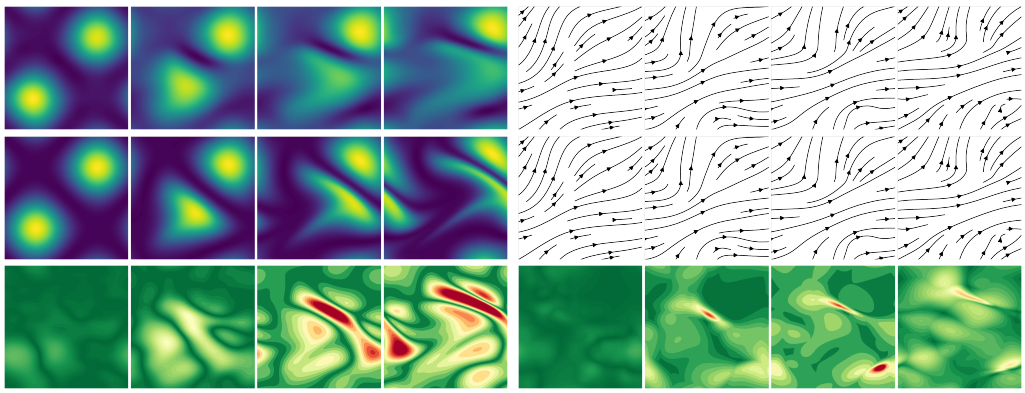
where N is chosen as 8, and θ is the standard embedding [Do Carmo, 1998], scaled by the frequency 2^k. The continuity equation is enforced exactly, using the method detailed in 3.2, so the L_{Cont} term is omitted from L_{total}. The initial velocity and density are

$$\rho_0(x,y) = (z_1 + z_3)^2 + 1 \qquad v_0(x,y) = [e^{z_3}, e^{z_1}/2]$$

where $z = \theta(x,y,1)$. Here we used the matrix-field parameterization as referred to in 3, as it is empirically more flexible in low dimensions. We also include the harmonic component from Hodge decomposition, which is just a constant vector field, to ensure we have full universality. Full methodology for the experiment is detailed in Appendix C. The results of evolving the flow for time $t \in [0, 1/3]$ are shown in Figure 2, as well as a comparison with a reference finite element solution generated by the conservative splitting method from [Guermond and Quartapelle, 2000]. In general, our model fits the initial conditions well and results in fluid-like movement of the density, however small approximation errors can accumulate from simulating over a longer time horizon. The fluid equations can be quite difficult to fit to and using adding more specialized methods such as those in Rao et al. [2020] can improve the result.

6 Dynamical Optimal Transport

The calculation of optimal transport (OT) maps between two densities of interest p_0 and p_1 serves as a powerful tool for computing distances between distributions. The inclusion of OT into machine learning has inspired multiple works on building efficient or stable generative models [Arjovsky et al., 2017, Tong et al., 2020, Makkuva et al., 2020, Huang et al., 2020, Finlay et al., 2020, Rout et al., 2021, Rozen et al., 2021]. In particular, we are interested in the dynamical formulation of Benamou and Brenier [2000], which is a specific instance where the transport map is characterized through a



(a) Density (b) Velocity

Figure 2: Our method (*top*) vs a reference FEM solution (*middle*), and an absolute error plot between the two (*bottom*). The simulation's error accumulates away from the initial value due to a slight mismatch in the fluid equations, but the general behavior has been learned.

vector field. Specifically, we wish to solve the following optimization problem

$$\min_{\rho,u} \int_0^1 \int_{\mathcal{M}} \|u(t,x)\|^2 \rho(t,x)\, dxdt \tag{18}$$

subject to the boundary constraints, $\rho(0,\cdot) = p_0(\cdot)$ and $\rho(1,\cdot) = p_1(\cdot)$, as well as remaining non-negative $\rho \geq 0$ on the domain and satisfying the continuity equation $\frac{\partial \rho}{\partial t} + \text{div}(\rho u) = 0$. In many existing cases, the models for estimating either ρ or eq. (18) are completely decoupled from the generative model or transport map [Arjovsky et al., 2017, Makkuva et al., 2020], or in some cases ρ would be computed through numerical simulation and eq. (18) estimated through samples [Finlay et al., 2020, Tong et al., 2020]. These approaches may not be ideal as they require an auxiliary process to satisfy or approximate the continuity equation.

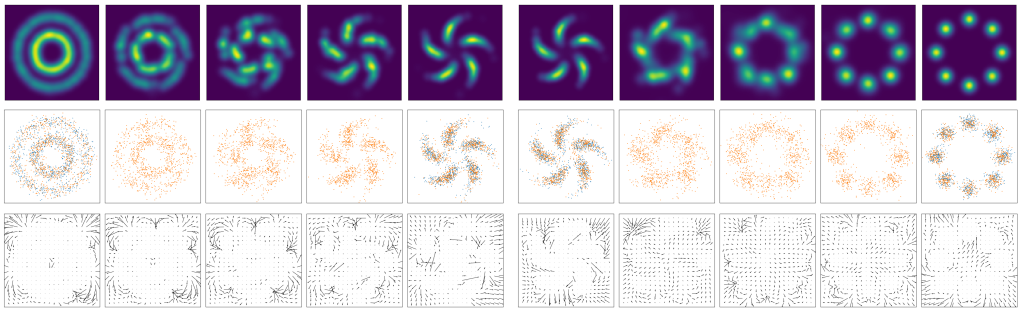
We consider the case where we have access to ρ and u that always satisfies the continuity equation, without the need for any numerical simulation. Instead, we only need to match the boundary conditions and optimize u with respect to the objective eq. (18).

Parameterizing densities with subharmonic functions In this setting, it is important that our density is non-negative for the entire domain and within the time interval between 0 and 1. Instead of adding another loss function to satisfy this condition, we show how we can easily bake this condition into our model using the implicit formulation of our divergence-free neural network.

Recall for the vector-field parameterization of the continuity equation, we have a vector field $b_\theta(y)$ with $y = [t, x]$, for $t \in [0,1]$, $x \in \mathbb{R}^n$. This results in the parameterization $\rho = \text{div}(\frac{\partial b_0}{\partial x} - \frac{\partial b_{1:n}}{\partial t})$ for $i = 0, \ldots, n$. We now make the adjustment that $\frac{\partial b_{1:n}}{\partial t} = 0$. This condition does not reduce the expressivity of the formulation, as ρ can still represent any density function. This modification does imply that $\rho = \nabla^2 b_0$ with ∇^2 being the Laplacian operator. Interestingly, this means that satisfying the boundary conditions is equivalent to solving a time-dependent Poisson's equation $\nabla^2 b_0(t,\cdot) = p_t$ for $t \in \{0,1\}$. Furthermore, the class of subharmonic functions is exactly the set of functions g that satisfies $\nabla^2 g(0,\cdot) \geq 0$. In one dimension, this corresponds to the class of convex functions but is a strict generalization of convex functions in higher dimensions. With this in mind, we propose using the parameterization

$$b_0(t,x) = \sum_{k=1}^{K} w_k(t)\phi(a_k(t)x + b_k(t)) \quad \text{where } \phi(z) = \tfrac{1}{4\pi}\left(\log(\|z\|^2) + E_1(\|z\|^2/4)\right) \tag{19}$$

with E_1 being the exponential integral function, and $w_k, a_k \in \mathbb{R}^+$ and $b_k \in \mathbb{R}^n$ are free functions of t. This a weighted sum of generalized linear models, where the choice of nonlinear function ϕ is chosen because it is the exact solution to the Poisson's equation for a standard normal distribution when $n = 2$, *i.e.* $\nabla^2 \phi = \mathcal{N}(0, I)$, while for $n \geq 2$ this is a subharmonic function as long as the free



(a) Circles $\longleftrightarrow$ Pinwheel (b) Pinwheel $\longleftrightarrow$ 8Gaussians

Figure 3: Learned approximation to the dynamical optimal transport map. (*top*) Density ρ. (*mid*) Transformed samples in orange, with samples from p_0 and p_1 in blue. (*bottom*) Learned vector field.

parameters are positive. As such, this can be seen as a generalization of a mixture model, which are often referenced to be universal density approximators [Goodfellow et al., 2016], albeit requiring many mixture components.

Experiments We experiment with pairs of 2D densities (visualized in Figure 3) and fit a pair of (ρ, u) to equation 18 while satisfying boundary conditions. Specifically, we train with the loss function

$$\min_{\rho,u} \; \mathbb{E}_{x\sim\tilde{p}_0}\left[|\rho(0,x) - p_0(x)|\right] + \mathbb{E}_{x\sim\tilde{p}_1}\left[|\rho(1,x) - p_1(x)|\right] + \lambda_0 \int_0^1 \int_{\mathcal{M}} \|u(t,x)\|^2 \rho(t,x)\, dxdt \tag{20}$$

where λ was tuned as a hyperparameter in $\{0.001, 0.005, 0.01\}$, and $\tilde{p}_i$ are mixture models of p_i and a uniform density over a sufficiently large area. We use a batch size of 64 and set K=128 (from equation 19). Qualitative results are displayed in Figure 3, which include interpolations between pairs of densities.

Furthermore, we can use the trained models to estimate the Wasserstein distance, by transforming samples from p_0 through the learned vector field to time t=1. We do this for 2000 samples and compare the estimated value to one estimated by that of a reference baseline optimal transport solver [Bonneel et al., 2011] based on a kernel density approximation, interfaced through the `pot` library [Flamary et al., 2021]. We estimate the Wasserstein distance multiple times and report the average and two standard deviations in Table 1. We see that the estimated values largely agree with the reference algorithm.

Table 1: Estimated Wasserstein distances.

	NCL (Ours)	Reference
Circles $\leftrightarrow$ Pinwheel	$0.086_{\pm 0.007}$	$0.089_{\pm 0.007}$
Pinwheel $\leftrightarrow$ 8Gaussians	$0.273_{\pm 0.003}$	$0.270_{\pm 0.008}$
8Gaussians $\leftrightarrow$ Circles	$0.182_{\pm 0.008}$	$0.184_{\pm 0.003}$

7 Conclusion

We proposed two constructions for building divergence-free neural networks from an original unconstrained smooth neural network. This resulted in two practical approaches where either an original matrix field or vector field is parameterized. We found both to be useful in practice; the matrix field approach is more flexible in lower dimensions while the vector field approach allows the addition of extra constraints such as a positive constraint. We combine these methods with the insight that the continuity equation itself is a divergence-free vector field. This resulted in a method for building deep neural networks that are constrained to always satisfy the continuity equation, which we refer to as Neural Conservation Laws. Currently, these models are difficult to scale due to extensive use of automatic differentiation for computing divergence of neural networks. It may be possible to combine our approach with parameterizations that provide cheap divergence computations [Chen and Duvenaud, 2019]. Nevertheless, the development of efficient batching and automatic differentiation tools is an active research area, so we expect these tools to improve as development progresses.

References

A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. A Conservative Adaptive Projection Method for the Variable Density Incompressible Navier–Stokes Equations. *Journal of Computational Physics*, 142(1):1–46, May 1998. ISSN 00219991. doi:10.1006/jcph.1998.5890. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999198958909.

B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

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

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

G. Arvanitidis, L. K. Hansen, and S. Hauberg. Latent Space Oddity: on the Curvature of Deep Generative Models, Dec. 2021. URL http://arxiv.org/abs/1710.11379. Number: arXiv:1710.11379 arXiv:1710.11379 [stat].

S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

C. Barbarosie. Representation of divergence-free vector fields. *Quarterly of applied mathematics*, 69 (2):309–316, 2011.

J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

M. Berger. A panoramic view of riemannian geometry. 2003.

N. Bonneel, M. Van De Panne, S. Paris, and W. Heidrich. Displacement interpolation using lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*, pages 1–12, 2011.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

R. T. Q. Chen and D. K. Duvenaud. Neural networks with cheap differential operators. *Advances in Neural Information Processing Systems*, 32, 2019.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

R. T. Q. Chen, J. Behrmann, D. K. Duvenaud, and J.-H. Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

M. P. Do Carmo. *Differential forms and applications*. Springer Science & Business Media, 1998.

M. Eisenberger, Z. Lähner, and D. Cremers. Divergence-Free Shape Interpolation and Correspondence, Oct. 2018. URL http://arxiv.org/abs/1806.10417. Number: arXiv:1806.10417 arXiv:1806.10417 [cs].

R. Feynman, R. Leighton, and M. Sands. *The Feynman Lectures on Physics: Commemorative Issue*. Advanced book program. Addison-Wesley, 1989. ISBN 978-0-201-50064-6. URL https://books.google.ca/books?id=UHp1AQAACAAJ.

C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. M. Oberman. How to train your neural ode. *arXiv preprint arXiv:2002.02798*, 2020.

R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

J. E. Gerken, J. Aronsson, O. Carlsson, H. Linander, F. Ohlsson, C. Petersson, and D. Persson. Geometric deep learning and equivariant neural networks, 2021. URL https://arxiv.org/abs/2105.13926.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

J.-L. Guermond and L. Quartapelle. A Projection FEM for Variable Density Incompressible Flows. *Journal of Computational Physics*, 165(1):167–188, Nov. 2000. ISSN 00219991. doi:10.1006/jcph.2000.6609. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999100966099.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

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

A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, June 2020. ISSN 00457825. doi:10.1016/j.cma.2020.113028. URL https://linkinghub.elsevier.com/retrieve/pii/S0045782520302127.

X. Jin, S. Cai, H. Li, and G. E. Karniadakis. NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, Feb. 2021. ISSN 00219991. doi:10.1016/j.jcp.2020.109951. URL http://arxiv.org/abs/2003.06496. arXiv: 2003.06496.

J. Kelliher. Stream functions for divergence-free vector fields. *Quarterly of Applied Mathematics*, 79 (1):163–174, 2021.

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(5):987–1000, 1998.

C. Li, W. Wei, J. Li, J. Yao, X. Zeng, and Z. Lv. 3dmol-net: learn 3d molecular representation using adaptive graph convolutional network based on rotation invariance. *IEEE Journal of Biomedical and Health Informatics*, 2021.

C. Lu, J. Chen, C. Li, Q. Wang, and J. Zhu. Implicit normalizing flows. *arXiv preprint arXiv:2103.09527*, 2021.

A. Makkuva, A. Taghvaei, S. Oh, and J. Lee. Optimal transport mapping via input convex neural networks. In *International Conference on Machine Learning*, pages 6672–6681. PMLR, 2020.

Z. Mao, A. D. Jagtap, and G. E. Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, Mar. 2020. ISSN 00457825. doi:10.1016/j.cma.2019.112789. URL https://linkinghub.elsevier.com/retrieve/pii/S0045782519306814.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

S. Morita. *Geometry of differential forms*. Number 201. American Mathematical Soc., 2001.

D. Onken, S. Wu Fung, X. Li, and L. Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, Nov. 2017. URL http://arxiv.org/abs/1711.10561. Number: arXiv:1711.10561 arXiv:1711.10561 [cs, math, stat].

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi:https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

C. Rao, H. Sun, and Y. Liu. Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 10(3):207–212, 2020.

L. Rout, A. Korotin, and E. Burnaev. Generative modeling with optimal transport maps. *arXiv preprint arXiv:2110.02999*, 2021.

N. Rozen, A. Grover, M. Nickel, and Y. Lipman. Moser flow: Divergence-based generative modeling on manifolds. *Advances in Neural Information Processing Systems*, 34, 2021.

P. W. Schroeder and G. Lube. Pressure-robust analysis of divergence-free and conforming FEM for evolutionary incompressible navier–stokes flows. *Journal of Numerical Mathematics*, 25(4), dec 2017. doi:10.1515/jnma-2016-1101. URL https://doi.org/10.1515%2Fjnma-2016-1101.

A. Tong, J. Huang, G. Wolf, D. Van Dijk, and S. Krishnaswamy. Trajectorynet: A dynamic optimal transport network for modeling cellular dynamics. In *International Conference on Machine Learning*, pages 9526–9536. PMLR, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In the conclusion section.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We don't foresee any.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Main results should be reproducible.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]