# Tractable Density Estimation on Learned Manifolds with Conformal Embedding Flows

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Normalizing flows are generative models that provide tractable density estimation by transforming a simple base distribution into a complex target distribution. However, this technique cannot directly model data supported on an unknown low-dimensional manifold, a common occurrence in real-world domains such as image data. Recent attempts to remedy this limitation have introduced geometric complications that defeat a central benefit of normalizing flows: exact density estimation. We recover this benefit with Conformal Embedding Flows, a framework for designing flows that learn manifolds with tractable densities. We argue that composing a standard flow with a trainable conformal embedding is the most natural way to model manifold-supported data. To this end, we present a series of conformal building blocks and apply them in experiments with real-world and synthetic data to demonstrate that flows can model manifold-supported distributions without sacrificing tractable likelihoods.

## 1 Introduction

Deep generative modelling is a rapidly evolving area of research in which the goal is to model a complex probability distribution from a set of samples. Research has been split between major approaches such as normalizing flows (NFs) [10, 41], generative adversarial networks (GANs) [15], variational autoencoders (VAEs) [25], energy-based models [12] and autoregressive models [35]. NFs in particular describe a distribution by modelling a change of variable function to a known base density. This approach provides the unique combination of efficient inference, efficient sampling, and exact likelihood estimation, but in practice generated images have not been as detailed or realistic as those of those of other methods [4, 22, 45, 19, 5].

One limiting aspect of traditional NFs is the use of a base density that has the same dimensionality as the data. This stands in contrast to models such as GANs and VAEs, which generate data by sampling from a low-dimensional latent prior and mapping the sample to data space. In many application domains, it is known or commonly assumed that the data of interest lives on a lower dimensional manifold embedded in the higher dimensional data space [13]. For example when modelling images, data samples belong to $[0, 1]^n$, where $n$ is the number of pixels in each image and each pixel has a brightness in the domain $[0, 1]$. However, most points in this data space correspond to meaningless noise, whereas meaningful images of objects lie on an intrinsic manifold of dimension $m \ll n$. A traditional NF cannot take advantage of the lower dimensional nature of realistic images.

There is growing research interest in modifying flow-based generative models to incorporate a change in dimensionality between the base density and data space [3, 7, 28, 27, 6]. Flows from low-dimensional latent spaces to high-dimensional data spaces could benefit from making better use of fewer parameters, being more memory efficient, and could reveal information about the intrinsic structure of the data. Properties of the data manifold can be of interest as well, such as its

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

dimensionality, or the semantic meaning of latent directions [25, 40]. However, leading approaches to manifold-based flows still suffer from drawbacks including intractable density estimation [3] or reliance on stochastic inverses [7].

In this paper we propose Conformal Embedding Flows (CEFs), a class of flows that use *conformal embeddings* to transform from low to high dimensions while maintaining invertibility and an efficiently computable density. We show how conformal embeddings can be used to learn a lower dimensional data manifold, and we combine them with powerful NF architectures for learning densities. The overall CEF paradigm permits efficient density estimation, sampling, and inference. We propose several types of conformal embedding that can be implemented as composable layers of a flow, including a new invertible layer, the orthogonal $k \times k$ convolution. Lastly, we demonstrate their efficacy on synthetic and real-world data.

# 2 Background

## 2.1 Normalizing Flows

In the traditional setting of a normalizing flow [10, 41], a set of data points $\{\mathbf{x}_i\} \subset \mathcal{X} = \mathbb{R}^n$ treated as random samples from an unknown density $p^*_{\mathbf{x}}(\mathbf{x})$ is used to learn an approximate density $p_{\mathbf{x}}(\mathbf{x})$ via maximum likelihood estimation. The approximate density is modelled using a diffeomorphism $\mathbf{f} : \mathcal{Z} \to \mathcal{X}$ which maps a base density $p_{\mathbf{z}}(\mathbf{z})$ over the space $\mathcal{Z} = \mathbb{R}^n$, typically taken to be a multivariate normal, to $p_{\mathbf{x}}(\mathbf{x})$ via the change of variables formula

$$p_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{z}}\left(\mathbf{f}^{-1}(\mathbf{x})\right) \left|\det \mathbf{J}_{\mathbf{f}}\left(\mathbf{f}^{-1}(\mathbf{x})\right)\right|^{-1}, \tag{1}$$

where $\mathbf{J}_{\mathbf{f}}(\mathbf{z})$ is the Jacobian matrix of $\mathbf{f}$ at the point $\mathbf{z}$. The change of variables formula derives from the principle of conservation of probability: the probability mass in an infinitesimal volume $\mathbf{dz}$ of $\mathcal{Z}$ must be preserved in the volume of $\mathcal{X}$ corresponding to the image $\mathbf{f}(\mathbf{dz})$, and it is exactly the Jacobian determinant that accounts for changes in the coordinate volume induced by $\mathbf{f}$. By parameterizing classes of diffeomorphisms $\mathbf{f}_\theta$, the mapping can be learned with the maximum likelihood loss function for the sample set $\{\mathbf{x}_i\} \subset \mathcal{X}$. Altogether, the three following operations must be tractable: sampling with $\mathbf{f}(\mathbf{z})$, inference with $\mathbf{f}^{-1}(\mathbf{x})$, and density estimation with the $|\det \mathbf{J}_{\mathbf{f}}(\mathbf{z})|$ factor. To scale the model, one can compose many such layers $\mathbf{f} = \mathbf{f}_k \circ \cdots \circ \mathbf{f}_1$, and the $|\det \mathbf{J}_{\mathbf{f}_i}(\mathbf{z})|$ factors multiply in Eq. (1).

Generally, there is no unifying way to parameterize an arbitrary bijection satisfying these constraints. Instead, normalizing flow research has progressed by designing new and more expressive component bijections which can be parameterized, learned, and composed. In particular, progress has been made by designing layers whose Jacobian determinants are tractable by construction. A significant theme has been to structure flows to have a triangular Jacobian [10, 41, 26, 11]. Kingma and Dhariwal [24] introduced invertible $1 \times 1$ convolution layers for image modelling; these produce block-diagonal Jacobians whose blocks are parameterized in a $PLU$-decomposition, so that the determinant can be computed in $\mathcal{O}(c)$, the number of input channels. See [37] for a thorough survey of normalizing flow models.

## 2.2 Injective Flows

The requirement that $\mathbf{f}$ be a diffeomorphism fixes the dimensionality of the latent space. In turn, $p_{\mathbf{x}}(\mathbf{x})$ must have full support over $\mathcal{X}$, which is problematic when the data lies on a submanifold $\mathcal{M} \subset \mathcal{X}$ with dimension $m < n$. Dai and Wipf [8] observed that if a probability model with full support is fitted via maximum likelihood to such data, the estimated density can converge towards infinity on $\mathcal{M}$ while ignoring the probability distribution of the data entirely. Behrmann et al. [2] point out that invertible neural networks can become numerically non-invertible, especially when the effective dimensionality of data and latents are mismatched. Correctly learning the data manifold along with its density may be able to circumvent these pathologies.

Injective flows seek to learn an explicitly low-dimensional support by reducing the dimensionality of the latent space and modelling the flow as a *smooth embedding*, or an injective function which is

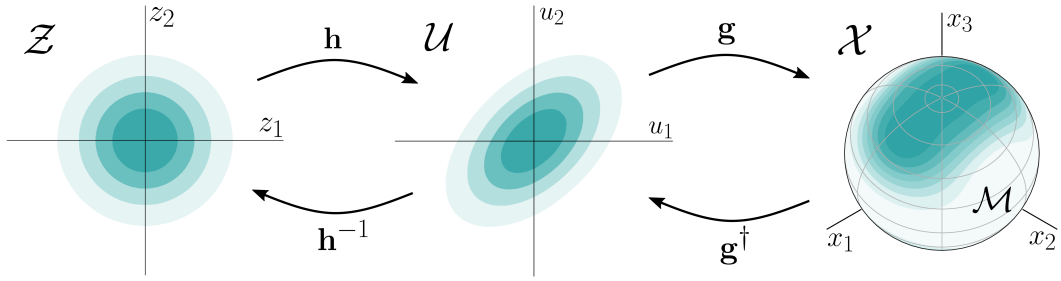



Figure 1: A normalized base density in the $\mathcal{Z}$ space is mapped by a bijective flow $\mathbf{h}$ to a more complicated density in $\mathcal{U}$. The injective component $\mathbf{g}$ maps this density onto a manifold $\mathcal{M}$ in $\mathcal{X}$. For inference, data points from $\mathcal{M}$ follow the reverse path through $\mathbf{g}^\dagger$ and $\mathbf{h}^{-1}$ to the latent space where their likelihood can be evaluated, and combined with the determinant factors in Eq. (4).

diffeomorphic to its image[1]. This case can be accommodated with a generalized change of variables formula for densities as follows [14].

Let $\mathbf{g} : \mathcal{U} \to \mathcal{X}$ be a smooth embedding from a latent space $\mathcal{U}$ onto the data manifold $\mathcal{M} \subset \mathcal{X}$, that is, $\mathcal{M} = \mathbf{g}(\mathcal{U})$ is the range of $\mathbf{g}$. Since $\mathbf{g}$ is a smooth embedding it has a left-inverse[2] $\mathbf{g}^\dagger : \mathcal{X} \to \mathcal{U}$ which is smooth on $\mathcal{M}$ and satisfies $\mathbf{g}^\dagger(\mathbf{g}(\mathbf{u})) = \mathbf{u}$ for all $\mathbf{u} \in \mathcal{U}$. Suppose $p_\mathbf{u}(\mathbf{u})$ is a density on $\mathcal{U}$ described using coordinates $\mathbf{u}$. The same density can be described in the ambient space $\mathcal{X}$ using $\mathbf{x}$ coordinates by pushing it through $\mathbf{g}$.

The quantity that accounts for changes in the coordinate volume at each point $\mathbf{u}$ is $\sqrt{\det\left[\mathbf{J}_\mathbf{g}^T(\mathbf{u})\mathbf{J}_\mathbf{g}(\mathbf{u})\right]}$, where the Jacobian $\mathbf{J}_\mathbf{g}$ is now a $n \times m$ matrix [29]. Hence, using the shorthand $\mathbf{u} = \mathbf{g}^\dagger(\mathbf{x})$, the generalized change of variables formula defined for $\mathbf{x} \in \mathcal{M}$ can be written

$$p_\mathbf{x}(\mathbf{x}) = p_\mathbf{u}(\mathbf{u}) \left|\det\left[\mathbf{J}_\mathbf{g}^T(\mathbf{u})\mathbf{J}_\mathbf{g}(\mathbf{u})\right]\right|^{-\frac{1}{2}}. \tag{2}$$

While $\mathbf{g}$ describes how the data manifold $\mathcal{M}$ is embedded in the larger ambient space, the mapping $\mathbf{g}$ alone may be insufficient to represent a normalized base density. As before, it is helpful to introduce a latent space $\mathcal{Z}$ of dimension $m$ along with a diffeomorphism $\mathbf{h} : \mathcal{Z} \to \mathcal{U}$ representing a bijective NF between $\mathcal{Z}$ and $\mathcal{U}$ [3]. Taking the overall injective transformation $\mathbf{g} \circ \mathbf{h}$ and applying the chain rule $\mathbf{J}_{\mathbf{g}\circ\mathbf{h}} = \mathbf{J}_\mathbf{g}\mathbf{J}_\mathbf{h}$ simplifies the determinant in Eq. (2) since the outer Jacobian $\mathbf{J}_\mathbf{h}$ is square,

$$\det\left[\mathbf{J}_\mathbf{h}^T\mathbf{J}_\mathbf{g}^T\mathbf{J}_\mathbf{g}\mathbf{J}_\mathbf{h}\right] = (\det \mathbf{J}_\mathbf{h})^2 \det\left[\mathbf{J}_\mathbf{g}^T\mathbf{J}_\mathbf{g}\right]. \tag{3}$$

Finally, writing $\mathbf{z} = \mathbf{h}^{-1}(\mathbf{u})$, the data density is modelled by

$$p_\mathbf{x}(\mathbf{x}) = p_\mathbf{z}(\mathbf{z}) \left|\det \mathbf{J}_\mathbf{h}(\mathbf{z})\right|^{-1} \left|\det\left[\mathbf{J}_\mathbf{g}^T(\mathbf{u})\mathbf{J}_\mathbf{g}(\mathbf{u})\right]\right|^{-\frac{1}{2}}, \tag{4}$$

with the entire process depicted in Fig. 1.

Generating samples from $p_\mathbf{x}(\mathbf{x})$ is simple; a sample $\mathbf{z} \sim p_\mathbf{z}(\mathbf{z})$ is drawn from the base density and passed through $\mathbf{g} \circ \mathbf{h}$. Inference on a data sample $\mathbf{x} \sim p_\mathbf{x}(\mathbf{x})$ is achieved by passing it through $\mathbf{h}^{-1} \circ \mathbf{g}^\dagger$, evaluating the likelihood according to $p_\mathbf{z}(\mathbf{z})$, and computing both determinant factors.

Notably, the modeled density $p_\mathbf{x}(\mathbf{x})$ only has support on a low dimensional subset of $\mathcal{X}$, as per the manifold assumption for realistic data. In practice, though, there will be off-manifold points during training or if $g(\mathbf{u})$ cannot perfectly fit the data, in which case the model's log-likelihood will be $-\infty$. Cunningham et al. [7] remedy this by adding an off-manifold noise term to the model, but inference requires a stochastic inverse, and the model must be optimized using an ELBO-like objective. Other

[1] Throughout this work we use the term "embedding" in the topological sense of a function describing how a low-dimensional space sits as a manifold inside a high-dimensional space. This is not to be confused with other uses for the term in machine learning, namely a low-dimensional representation of high-dimensional or discrete data.

[2] $\dagger$ denotes a left-inverse function, not necessarily the matrix pseudoinverse.

work [3, 27] has projected data to the manifold via $\mathbf{g} \circ \mathbf{g}^\dagger$ prior to computing log-likelihoods, and optimized $\mathbf{g}$ using the reconstruction loss $\mathbb{E}_{\mathbf{x}\sim p^*_{\mathbf{x}}}||\mathbf{x} - \mathbf{g}(\mathbf{g}^\dagger(\mathbf{x}))||^2$.

When computing log-likelihoods, the determinant term $\log\det\left[\mathbf{J}_{\mathbf{g}}^T\mathbf{J}_{\mathbf{g}}\right]$ presents a computational challenge. Kumar et al. [28] maximize it using an approximate lower bound, while Brehmer and Cranmer [3] and Kothari et al. [27] circumvent its computation altogether by only maximizing the other terms in the log-likelihood. In line with research to build expressive bijective flows where $\det \mathbf{J}_{\mathbf{f}}$ is tractable, our work focuses on designing and parameterizing injective flows where $\log\det\left[\mathbf{J}_{\mathbf{g}}^T\mathbf{J}_{\mathbf{g}}\right]$ as a whole is efficiently computable. In contrast to past injective flow models, our approach allows for straightforward evaluation and optimization of $\log\det\left[\mathbf{J}_{\mathbf{g}}^T\mathbf{J}_{\mathbf{g}}\right]$ in the same way standard NFs do for $\log|\det \mathbf{J}_{\mathbf{f}}|$. As far as we can find, ours is the first approach to make this task tractable at scale.

# 3 Conformal Embedding Flows

In this section we propose Conformal Embedding Flows (CEFs) as a method for learning the low-dimensional manifold $\mathcal{M} \subset \mathcal{X}$ and the probability density of the data on the manifold.

Modern bijective flow work has produced tractable $\log|\det \mathbf{J}_{\mathbf{f}}|$ terms by designing layers with triangular Jacobians[10, 11]. For injective flows, the combination $\mathbf{J}_{\mathbf{g}}^T\mathbf{J}_{\mathbf{g}}$ is symmetric, so it is triangular if and only if it is diagonal. In turn, $\mathbf{J}_{\mathbf{g}}^T\mathbf{J}_{\mathbf{g}}$ being diagonal is equivalent to $\mathbf{J}_{\mathbf{g}}$ having orthogonal columns. While this restriction is feasible for a single layer $\mathbf{g}$, it is not composable. If $\mathbf{g}_1$ and $\mathbf{g}_2$ are both smooth embeddings whose Jacobians have orthogonal columns, it need not follow that $\mathbf{J}_{\mathbf{g}_2\circ\mathbf{g}_1}$ has orthogonal columns. Additionally, since the Jacobians are not square, $\det\left[\mathbf{J}_{\mathbf{g}_1}^T\mathbf{J}_{\mathbf{g}_2}^T\mathbf{J}_{\mathbf{g}_2}\mathbf{J}_{\mathbf{g}_1}\right]$, the determinant in Eq. (2), cannot be factored into a product of individually computable terms as in Eq. (3). To ensure composability we propose enforcing the slightly more strict criteria that each $\mathbf{J}_{\mathbf{g}}^T\mathbf{J}_{\mathbf{g}}$ be a scalar multiple of the identity. This is precisely the condition that $\mathbf{g}$ is a conformal embedding.

Formally, $\mathbf{g} : \mathcal{U} \to \mathcal{X}$ is a conformal embedding if it is a smooth embedding whose Jacobian satisfies

$$\mathbf{J}_{\mathbf{g}}^T(\mathbf{u})\mathbf{J}_{\mathbf{g}}(\mathbf{u}) = \lambda^2(\mathbf{u})\mathbf{I}_m\,, \tag{5}$$

where $\lambda : \mathcal{U} \to \mathbb{R}$ is a smooth non-zero scalar function, the *conformal factor* [29]. In other words, $\mathbf{J}_{\mathbf{g}}$ has orthonormal columns up to a non-zero multiplicative constant. Hence $\mathbf{g}$ locally preserves angles.

From Eq. (5) it is clear that conformal embeddings naturally satisfy our requirements as an injective flow. In particular, let $\mathbf{g} : \mathcal{U} \to \mathcal{X}$ be a conformal embedding and $\mathbf{h} : \mathcal{Z} \to \mathcal{U}$ be a standard normalizing flow model. The injective flow model $\mathbf{g} \circ \mathbf{h} : \mathcal{Z} \to \mathcal{X}$ satisfies

$$p_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{z}}(\mathbf{z})\left|\det \mathbf{J}_{\mathbf{h}}(\mathbf{u})\right|^{-1}\lambda^{-m}(\mathbf{u})\,. \tag{6}$$

We call $\mathbf{g} \circ \mathbf{h}$ a Conformal Embedding Flow.

There are multiple options for training a CEF. End-to-end maximum likelihood training could be used for $\mathbf{g} \circ \mathbf{h}$ as a whole, but when projecting to the manifold, it is possible to maximize density without learning the manifold correctly [3]. Without any initialization of $\mathbf{g}$ the model will struggle to isolate the data manifold $\mathcal{M}$ as the range of $\mathbf{g}$. We find it effective to first train the manifold learner $\mathbf{g}$ alone for several epochs by minimizing the reconstruction loss $\mathbb{E}_{\mathbf{x}\sim p^*_{\mathbf{x}}}\|\mathbf{x} - \mathbf{g}(\mathbf{g}^\dagger(\mathbf{x}))\|^2$. After this manifold warmup phase, our model density can be optimized in two ways. The first way is the *sequential* training approach, in which we optimize $\mathbf{h}$ with $\mathbf{g}$ fixed as in Brehmer and Cranmer [3]. The alternative, which we refer to as the *joint* training approach, is to optimize the following loss,

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}\sim p^*_{\mathbf{x}}}\left[-\log p_{\mathbf{x}}(\mathbf{x}) + \alpha\|\mathbf{x} - \mathbf{g}(\mathbf{g}^\dagger(\mathbf{x}))\|^2\right]. \tag{7}$$

This mixed loss is unique to our model because it is the first model for which $\log p_{\mathbf{x}}(\mathbf{x})$ is tractable.

## 3.1 Designing Conformal Embedding Flows

In general, there is no unifying way to parameterize all conformal embeddings (See App. A for more discussion). Despite this, we can identify expressive classes of embeddings. For a conformal embedding $\mathbf{g}$ to be useful in a CEF we must be able to sample with $\mathbf{g}(\mathbf{u})$, perform inference with $\mathbf{g}^\dagger(\mathbf{x})$, and compute the conformal factor $\lambda(\mathbf{u})$. To build a parameterizable and scalable model $\mathbf{g}$, it

helps to work with conformal building blocks $\mathbf{g}_i : \mathcal{U}_{i-1} \to \mathcal{U}_i$ (where $\mathcal{U}_0 = \mathcal{U}$ and $\mathcal{U}_k = \mathcal{X}$), which we compose to produce the full conformal embedding $\mathbf{g}$:

$$\mathbf{g} = \mathbf{g}_k \circ \cdots \circ \mathbf{g}_1 \,. \tag{8}$$

In turn, $\mathbf{g}$ is conformal because

$$\mathbf{J}_{\mathbf{g}}^T \mathbf{J}_{\mathbf{g}} = \left(\mathbf{J}_{\mathbf{g}_1}^T \cdots \mathbf{J}_{\mathbf{g}_k}^T\right)\left(\mathbf{J}_{\mathbf{g}_k} \cdots \mathbf{J}_{\mathbf{g}}\right) = \lambda_1^2 \cdots \lambda_k^2 \mathbf{I}_m \,. \tag{9}$$

Our goal in the remainder of this section is to design classes of conformal building blocks which can be parameterized and learned in a CEF.

### 3.1.1 Conformal Embeddings from Conformal Mappings

Consider the special case where the conformal embedding maps between Euclidean spaces $\mathcal{U} \subseteq \mathbb{R}^d$ and $\mathcal{V} \subseteq \mathbb{R}^d$ of the same dimension[3]. In this special case $\mathbf{g}_i$ is called a conformal mapping. Liouville's theorem [17] states that any conformal mapping can be expressed as a composition of translations, orthogonal transformations, scalings, and inversions, which are defined in Table 1 (see App. A.1 for details on conformal mappings). We created conformal embeddings primarily by composing these layers. Zero-padding (which is trivially conformal) was interspersed to provide changes in dimensionality [3].

Table 1: Conformal Mappings

| **Type** | **Functional Form** | **Inverse** | **Conformal Factor** $\lambda(\mathbf{u})$ |
|---|---|---|---|
| Translation | $\mathbf{u} \mapsto \mathbf{u} + \mathbf{a}, \quad \mathbf{a} \in \mathbb{R}^d$ | $\mathbf{v} \mapsto \mathbf{v} - \mathbf{a}$ | $1$ |
| Orthogonal | $\mathbf{u} \mapsto \mathbf{Q}\mathbf{u}, \quad \mathbf{Q} \in O(d)$ | $\mathbf{v} \mapsto \mathbf{Q}^T\mathbf{v}$ | $1$ |
| Scaling | $\mathbf{u} \mapsto \lambda\mathbf{u}, \quad \lambda \in \mathbb{R}$ | $\mathbf{v} \mapsto \lambda^{-1}\mathbf{v}$ | $\lambda$ |
| Inversion | $\mathbf{u} \mapsto \mathbf{u}/\lVert\mathbf{u}\rVert^2,$ | $\mathbf{v} \mapsto \mathbf{v}/\lVert\mathbf{v}\rVert^2$ | $\lVert\mathbf{u}\rVert^{-2}$ |
| SCT | $\mathbf{u} \mapsto \frac{\mathbf{u}-\lVert\mathbf{u}\rVert^2\mathbf{b}}{1-2\mathbf{b}\cdot\mathbf{u}+\lVert\mathbf{b}\rVert^2\lVert\mathbf{u}\rVert^2}, \ \mathbf{b} \in \mathbb{R}^d$ | $\mathbf{v} \mapsto \frac{\mathbf{v}+\lVert\mathbf{v}\rVert^2\mathbf{b}}{1+2\mathbf{b}\cdot\mathbf{v}+\lVert\mathbf{b}\rVert^2\lVert\mathbf{v}\rVert^2}$ | $1-2\mathbf{b}\cdot\mathbf{u}+\lVert\mathbf{b}\rVert^2\lVert\mathbf{u}\rVert^2$ |

Stacking translations, orthogonal transformations, scalings, and inversions layers is sufficient to learn any conformal mapping in principle. However, we found the inversion operation to be numerically unstable, so we supplemented it with the *special conformal transformation* (SCT), a transformation of interest in conformal field theory [9]. It can be understood as an inversion, followed by a translation by $\mathbf{b}$, followed by another inversion. In contrast to inversions, SCTs have a continuous parameter and include the identity when this parameter is set to 0.

The main challenge to achieving the full expressibility allowable within Liouville's theorem is implementing trainable orthogonal layers. We parameterized orthogonal transformations in two different ways: by using Householder matrices [44], which are cheaply parameterizable and easy to train, and by using the API provided by [30], which parameterizes arbitrary orthogonal matrices with determinant 1 as the matrix exponential of a skew-symmetric matrix.

To scale orthogonal transformations to image data, we propose a new invertible layer: the orthogonal $k \times k$ convolution. In the spirit of the invertible $1 \times 1$ convolutions of Kingma and Dhariwal [24], we note that a $k \times k$ convolution with stride $k$ has a block diagonal Jacobian. The Jacobian is orthogonal if and only if these blocks are orthogonal. It suffices then to convolve the input with a set of filters that together form an orthogonal matrix. It is also worth noting that these layers can be inverted efficiently by applying a transposed convolution with the same filter, while a standard invertible $1 \times 1$ convolution requires a matrix inversion. This facilitates quick forward and backward passes when optimizing the model's reconstruction loss.

### 3.1.2 Piecewise Conformal Embeddings

To increase expressivity of the embeddings, the conformality condition on $\mathbf{g}$ can be relaxed to the point of being conformal *almost everywhere*. Formally, the latent spaces $\mathcal{Z}$ and $\mathcal{U}$ are redefined

[3] We consider conformal mappings between spaces of dimension $d > 2$. Conformal mappings in $d = 2$ are much less constrained, while the case $d = 1$ is trivial since there is no notion of an angle.

as $\mathcal{Z} = \{\mathbf{z} : \mathbf{g} \text{ is conformal at } \mathbf{h}(\mathbf{z})\}$ and $\mathcal{U} = \mathbf{h}(\mathcal{Z})$. Then $\mathbf{g}$ remains a conformal embedding on $\mathcal{U}$, and as long as $\{\mathbf{x} : \mathbf{g} \text{ is nonconformal at } \mathbf{g}^\dagger(\mathbf{x})\}$ also has measure zero, this approach poses no practical problems. Note that the same relaxation is performed implicitly with the diffeomorphism property of standard flows when rectifier nonlinearities are used in coupling layers [11].

Table 2: Piecewise Conformal Embeddings

| **Type** | **Functional Form** | **Left-Inverse** | $\lambda(\mathbf{u})$ |
|---|---|---|---|
| Conformal ReLu [27] | $\mathbf{u} \mapsto \text{ReLU}\begin{bmatrix}\mathbf{Q}\mathbf{u}\\ -\mathbf{Q}\mathbf{u}\end{bmatrix}, \quad \mathbf{Q} \in O(d)$ | $\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\end{bmatrix} \mapsto \mathbf{Q}^T(\mathbf{v}_1 - \mathbf{v}_2)$ | 1 |
| Conditional Orthogonal | $\mathbf{u} \mapsto \begin{cases}\mathbf{Q}_1\mathbf{u} & \text{if } \lVert\mathbf{u}\rVert < 1\\ \mathbf{Q}_2\mathbf{u} & \text{if } \lVert\mathbf{u}\rVert \geq 1\end{cases}, \ \mathbf{Q}_1, \mathbf{Q}_2 \in O(d)$ | $\mathbf{v} \mapsto \begin{cases}\mathbf{Q}_1^T\mathbf{u} & \text{if } \lVert\mathbf{v}\rVert < 1\\ \mathbf{Q}_2^T\mathbf{u} & \text{if } \lVert\mathbf{v}\rVert \geq 1\end{cases}$ | 1 |

We considered the two piecewise conformal embeddings defined in Table 2. Due to the success of ReLU in standard deep neural networks [34], we try a ReLU-like layer that is piecewise conformal. *Conformal ReLU* is based on the injective ReLU proposed by Kothari et al. [27]. We believe it to be of general interest as a conformal nonlinearity, but it provided no performance improvements in our experiments. More useful was the *Conditional Orthogonal* transformation, which takes advantage of the norm-preservation of orthogonal transformations to create an invertible layer. Despite the Conditional Orthogonal layer being discontinuous, it provided a substantial boost in reconstruction ability on image data.

# 4 Related Work

**Flows on prescribed manifolds** Flows can be developed for Riemannian manifolds $\mathcal{M} \subseteq \mathcal{X}$ which are known in advance and can be defined as the image of some fixed $\phi : \mathcal{U} \to \mathcal{X}$, where $\mathcal{U} \subseteq \mathbb{R}^m$ [14, 33, 37]. In particular, Rezende et al. [42] model densities on spheres and tori with convex combinations of Möbius transformations, which are cognate to conformal mappings. For known manifolds $\phi$ is fixed, and the density's log Jacobian determinant term may be computable in closed form. Our work replaces $\phi$ with a trainable network $g$, but the log Jacobian determinant still has a simple closed form.

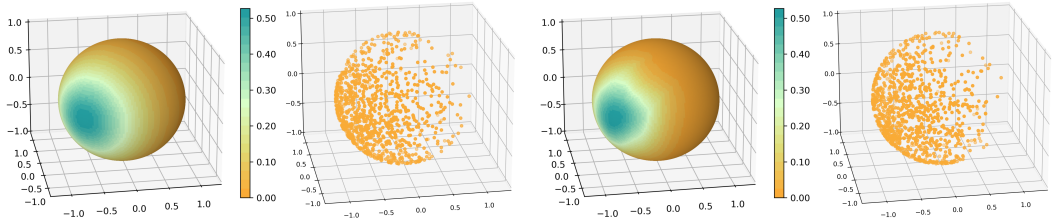
**Flows on learnable manifolds** Extending flows to learnable manifolds brings about two main challenges: handling off-manifold points, and training the density on the manifold.

If the manifold is the density's support, all off-manifold points during training will have a likelihood of 0. This has been addressed by adding an off-manifold noise term [7, 6] or by training the manifold with a reconstruction loss and projecting the data onto the manifold [28, 3, 27]. We opt for the latter approach.

Training the density on the manifold is challenging because the log-determinant term is typically intractable. Kumar et al. [28] use a series of lower bounds to train the log-determinant, while Brehmer and Cranmer [3] and Kothari et al. [27] separate the flow into two components, and only train the low-dimensional part of the flow. Our approach is the first injective model to provide a learnable manifold with exact log-determinant computation.

**Conformal, isometric, and orthogonal networks** Conformality has two special cases of interest: isometry and semi-orthogonality. An isometric embedding is a conformal embedding with a constant conformal factor of 1, while a semi-orthogonal transformation is a linear isometric embedding. Authors have typically imposed conformality [38], isometry [39, 46], or semi-orthogonality [1, 21] as a regularizer. These approaches have not enforced orthogonality strictly; the set of $n \times k$ semi-orthogonal matrices forms the *Stiefel manifold* in $\mathbb{R}^{n\times k}$, so strict orthogonality entails optimizing weights along this manifold.

Past work has trained along the Stiefel manifold in two ways: with Riemannian gradient descent, and by directly parametrizing the manifold. Riemannian gradient descent algorithms typically require a singular value decomposition or QR decomposition at each training step [16, 36, 20]. Lezcano-Casado [30] and Lezcano-Casado and Martínez-Rubio [31] parameterize the set of orthogonal matrices with positive determinant as the matrix exponential of a skew-symmetric matrix, wherein the matrix



(a) (b) (c) (d)

Figure 2: (a) A density $p^*_{\mathbf{x}}(\mathbf{x})$ with support on the sphere, and (b) $10^3$ samples comprising the training dataset $\{\mathbf{x_i}\}$. (c) The density learned by a CEF, and (d) $10^3$ generated samples.

exponential and its gradient must be approximated. Another approach is to directly parameterize a smaller subset of orthogonal matrices. Tomczak and Welling [44] construct a linear volume-preserving flow using $n \times n$ Householder matrices, which can be parameterized with $n - 1$ degrees of freedom. We found the matrix exponential and Householder approaches to be most efficient, so we used a mix of them.

# 5 Experiments

## 5.1 Synthetic Spherical Data

To demonstrate the how a CEF can jointly learn a manifold and density, we generated a synthetic dataset from a known distribution with support on a spherical surface embedded in $\mathbb{R}^3$ as described in App. B. The distribution visualized in Fig. 2, along with the training dataset of $10^3$ sampled points.

Since the manifold dimension is known, we constructed a simple $\mathbf{g}$ for learning the manifold composed from a padding layer, and the conformal mappings in Table 1. The bijective component $\mathbf{h}$ for density normalization consisted of three sets of Glow-style steps of {ActNorm, invertible $1 \times 1$ convolution, affine coupling} acting on the two latent dimensions as channels.

We trained the two components of the CEF jointly, with a reconstruction loss initialization phase for $\mathbf{g}$, followed by the mixed loss function in Eq. (7) with an end-to-end log-likelihood term. The resulting model density is plotted in Fig. 2 along with generated samples, and shows good fidelity to the known manifold and density.

## 5.2 Image Modelling

We scale CEFs to larger data by training on the CelebA dataset for which a low-dimension manifold structure is postulated but not known [32][4]. Our aim is to show that, despite restricting its functional form to provide exact likelihood estimation, CEFs are competitive with mainstream injective flow training approaches [3][27]. In doing so, we are the first to perform end-to-end maximum likelihood training with an injective flow on image data. Three approaches were evaluated: a sequentially trained baseline model, a sequentially trained CEF, and a jointly trained CEF. We fixed a small Glow-style stump with 3 levels and 3 steps per level as the backbone $\mathbf{h}$ for all models.

The baseline's embedding $\mathbf{g}$ is a Glow-style network of 3 levels and 2 steps per level: the output of each scale is reshaped into $8 \times 8$, and all scales are concatenated. We then apply an invertible $1 \times 1$ convolution, and project the input down to 1536 dimensions. Since this network is not conformal, joint training is intractable, so it must be trained sequentially. On the other hand, both CEFs use the same conformal architecture for $\mathbf{g}$: a series of $2 \times 2$ and $1 \times 1$ Householder convolutions interspersed with Conditional Orthogonal convolutions, padding layers, shifts, and scales. All models were trained with the Adam optimizer [23] with learning rate $1 \times 10^{-5}$ for 100 epochs. Using a single NVIDIA TITAN V, the sequential baseline and sequential CEF ran for 34 hours, while the joint CEF ran for 65 hours. More experimental details are provided in App. B.

[4]The CelebA dataset is available for non-commercial research purposes only. For specifics please refer to the CelebA website.

(a) (b) (c)

Figure 3: Generated samples (top) and reconstructions (bottom) for (a) the jointly trained CEF, (b) the sequentially trained CEF, and (c) the sequentially trained baseline model. In the reconstruction panes, original images are shown on the left, while reconstructions are shown on the right.

Injective models cannot be compared on the basis of log-likelihood, since each model may have a different manifold support. Instead, we compare images on the basis of FID score [18], of which the mean and standard deviation is computed for each model over 3 sets of generated samples. We use the popular PyTorch-FID package [43]. As expected, since the baseline embedding is more flexible, it achieves smaller reconstruction losses than the conformal models. However, the CEFs produce competitive FID scores, possibly due to the better global consistency of facial features from the two CEF models. In contrast, the baseline model learns locally sharp features, but more amorphous faces. We hypothesize that the conformally embedded manifold creates a simpler optimization surface, so the stump picks up on global symmetries more easily.

Table 3: CelebA Performance

| Approach | FID | Reconstruction | $\mathbf{g}$ + $\mathbf{h}$ Parameters |
|---|---|---|---|
| Sequential Baseline | $110.87 \pm 0.12$ | $0.402 \times 10^{-3}$ | 36346752 + 36429778 |
| Sequential CEF | $111.76 \pm 0.14$ | $1.046 \times 10^{-3}$ | 83036 + 36429778 |
| Joint CEF | $126.04 \pm 0.23$ | $1.043 \times 10^{-3}$ | 83036 + 36429778 |

## 6 Limitations and Future Directions

**Expressivity** Just as standard flows trade some expressibility for tractable likelihoods, so must injective flows. Our conformal networks in particular are less expressive than state-of-the-art flow models; for instance, our model produced blurry reconstructions of CelebA data when the latent dimension was below 1000, while Brehmer and Cranmer [3] use a latent dimension of 512. We mostly compose padding layers and dimension-*preserving* conformal mappings, which is a naturally restrictive class by Liouville's theorem [17]. Just as early work on NFs [41, 10] introduced limited classes of parameterizable bijections, which were later improved in many directions, our work introduces several classes of parameterizable conformal embeddings. We expect that future work will uncover more expressive dimension-*increasing* conformal embeddings.

**Manifold Learning** Strictly manifold-supported probability models such as ours introduce a bi-objective optimization problem. How to balance these objectives is unclear and, thus far, empirical [3]. The difference in supports between two manifold models also makes their likelihoods incomparable. Cunningham et al. [7] have made progress in this direction by adding noise to data generated on the manifold, but this makes inversion stochastic and introduces density estimation challenges. We suspect that using conformal manifold-learners may make density estimation more tractable in this setting, but further research is needed in this direction.

**Dataset bias and consent** Clearly, the CelebA dataset does not accurately represent the real-world distribution of human traits. Models trained on CelebA, and indeed any other possibly biased datasets, should be vetted for fairness before being deployed to make decisions that can adversely affect human beings. Intuitively, manifold-learning as a task should be robust to mild dataset biases, hence capturing underrepresented segments of a population better than standard generative models, but this has yet to be formalized or experimentally validated. We also note that while the human subjects in these images are all public figures, they may not have explicitly provided consent for their images to be used for research purposes.

# 7 Conclusion

In this paper, we introduced Conformal Embedding Flows for modelling probability distributions on low-dimensional manifolds while maintaining tractable densities. We showed that conformal embeddings naturally fit the framework of injective flows by providing fast sampling, invertibility for inference, and a simple Jacobian determinant factor for efficient likelihood estimation, all while being composable for scaling to deep models. Furthermore, the restrictions on conformal embeddings are minimal, in that any looser condition will sacrifice one or more of these properties. As we have reviewed, previous works on injective flows do not maintain all of these properties simultaneously.

Normalizing flows are still outperformed by other generative models such as GANs and VAEs in the arena of realistic image generation. Notably, these two alternatives benefit from a low-dimensional latent space, which better reflects image data's manifold structure and provides for more scalable model design. By equipping flows with a low-dimensional latent space, injective flow models have made progress towards achieving VAE- or GAN-level performance. The CEF paradigm is a way to match these strides while maintaining the theoretical strengths of NFs.

**Broader Impact** Natural image generation is a common task in generative modelling, with human face datasets frequently used as benchmarks for model performance. However, there are concerns that highly realistic generated images could be used for unethical purposes such as fraud. Our work is primarily focused on the theoretical properties of flow models, not on generating highly realistic images. As more powerful approaches already exist for realistic image generation [4, 22, 45, 5], our research is unlikely to lead to any broader impact on society in terms of unethical use of image generation.

# References


[1] N. Bansal, X. Chen, and Z. Wang. Can We Gain More from Orthogonality Regularizations in Training Deep Networks? In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[2] J. Behrmann, P. Vicol, K.-C. Wang, R. Grosse, and J.-H. Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pages 1792–1800. PMLR, 13–15 Apr 2021.

[3] J. Brehmer and K. Cranmer. Flows for simultaneous manifold learning and density estimation. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[4] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[5] R. Child. Very deep VAEs generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*, 2021.

[6] E. Cunningham and M. Fiterau. A change of variables method for rectangular matrix-vector products. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130. PMLR, 2021.

[7] E. Cunningham, R. Zabounidis, A. Agrawal, I. Fiterau, and D. Sheldon. Normalizing Flows Across Dimensions. *arXiv:2006.13070*, 2020.

[8] B. Dai and D. Wipf. Diagnosing and Enhancing VAE Models. In *International Conference on Learning Representations, ICLR 2019*, 2019.

[9] P. Di Francesco, P. Mathieu, and D. Sénéchal. *Conformal field theory*. Springer Science & Business Media, 2012.

[10] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv:1410.8516*, 2014.

[11] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations, ICLR 2017*, 2017.

[12] Y. Du and I. Mordatch. Implicit generation and modeling with energy based models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[13] C. Fefferman, S. Mitter, and H. Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[14] M. C. Gemici, D. Rezende, and S. Mohamed. Normalizing Flows on Riemannian Manifolds, 2016.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv:1406.2661*, 2014.

[16] M. Harandi and B. Fernando. Generalized BackPropagation, Étude De Cas: Orthogonality. *arXiv:1611.05927*, 2016.

[17] P. Hartman. On isometries and on a theorem of Liouville. *Mathematische Zeitschrift*, 69: 202–210, 1958.

[18] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[19] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[20] L. Huang, X. Liu, B. Lang, A. W. Yu, and B. Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *AAAI*, 2018.

[21] K. Jia, D. Tao, S. Gao, and X. Xu. Improving training of deep neural networks via singular value bounding. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4002, 2017.

[22] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. URL http://arxiv.org/abs/1412.6980.

[24] D. P. Kingma and P. Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[25] D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. *arXiv:1312.6114*, 2013.

[26] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[27] K. Kothari, A. Khorashadizadeh, M. de Hoop, and I. Dokmanić. Trumpets: Injective Flows for Inference and Inverse Problems. *arvix:2102.10461*, 2021.

[28] A. Kumar, B. Poole, and K. Murphy. Regularized Autoencoders via Relaxed Injective Probability Flow. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, pages 4292–4301, 2020.

[29] J. M. Lee. *Introduction to Riemannian manifolds*. Springer, 2018.

[30] M. Lezcano-Casado. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 9154–9164, 2019.

[31] M. Lezcano-Casado and D. Martínez-Rubio. Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3794–3803, 2019.

[32] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[33] E. Mathieu and M. Nickel. Riemannian Continuous Normalizing Flows. In *Advances in Neural Information Processing Systems*, volume 33, pages 2503–2515, 2020.

[34] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, 2010.

[35] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756. PMLR, 2016.

[36] M. Ozay and T. Okatani. Optimization on Submanifolds of Convolution Kernels in CNNs. *arXiv: 1610.07008*, 2016.

[37] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[38] E. Peterfreund, O. Lindenbaum, F. Dietrich, T. Bertalan, M. Gavish, I. G. Kevrekidis, and R. R. Coifman. Local conformal autoencoder for standardized data coordinates. *Proceedings of the National Academy of Sciences*, 117(49):30918–30927, 2020.

[39] H. Qi, C. You, X. Wang, Y. Ma, and J. Malik. Deep isometric learning for visual recognition. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 7824–7835, 2020.

[40] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *International Conference on Learning Representations*, 2016.

[41] D. Rezende and S. Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1530–1538, 2015.

[42] D. J. Rezende, G. Papamakarios, S. Racaniere, M. Albergo, G. Kanwar, P. Shanahan, and K. Cranmer. Normalizing Flows on Tori and Spheres. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 8083–8092, 2020.

[43] M. Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.1.1.

[44] J. M. Tomczak and M. Welling. Improving Variational Auto-Encoders using Householder Flow. *arXiv: 1611.09630*, 2016.

[45] A. Vahdat and J. Kautz. NVAE: A Deep Hierarchical Variational Autoencoder. In *Advances in Neural Information Processing Systems*, volume 33, pages 19667–19679, 2020.

[46] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5393–5402, 2018.

# 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] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec. 7.
   (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] Assumptions and theoretical results are explained in Sec. 2 and 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] Proofs of theoretical results are explained in Sec. 2 with additional information in App. A.
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)? [No] Experimental setups included but code will not be released until publication date.
   (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? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (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? [Yes] See Sec. 6.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Sec. 6.
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]