Abstract

Normalizing flows are generative models that provide tractable density estimation via an invertible transformation from a simple base distribution to 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 synthetic and real-world data to demonstrate that flows can model manifold-supported distributions without sacrificing tractable likelihoods.

1 Introduction

Deep generative modelling is the task of modelling a complex, high-dimensional data distribution from a sample set. Research has encompassed major approaches such as normalizing flows (NFs) [16,59], generative adversarial networks (GANs) [23], variational autoencoders (VAEs) [36], autoregressive models [52], energy-based models [18], score-based models [54], and diffusion models [29,63]. NFs in particular describe a distribution by modelling a change-of-variables mapping to a known base density. This approach provides the unique combination of efficient inference, efficient sampling, and exact density estimation, but in practice generated images have not been as detailed or realistic as those of other methods [7,10,29,52,63].

One limitation of traditional NFs is the use of a base density with 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 [21]. 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 a submanifold 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 injective flows, which account for unknown manifold structure by incorporating a base density of lower dimensionality than the data space \([0,1]^n\) [12,13,52,41]. Flows with low-dimensional latent 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, such as its dimensionality or the semantic meaning of latent directions,
can be of interest as well \cite{36,58}. However, leading injective flow models still suffer from drawbacks including intractable density estimation \cite{5} or reliance on stochastic inverses \cite{14}.

In this paper we propose Conformal Embedding Flows (CEFs), a class of flows that use \textit{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 three new invertible layers: the orthogonal \(k \times k\) convolution, the conditional orthogonal transformation, and the special conformal transformation. 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 \cite{15,59}, an independent and identically distributed sample \(\{x_i\} \subset \mathcal{X} = \mathbb{R}^n\) from an unknown ground-truth distribution with density \(p_x(x)\) is used to learn an approximate density \(p_x(x)\) via maximum likelihood estimation. The approximate density is modelled using a diffeomorphism \(f : \mathcal{Z} \rightarrow \mathcal{X}\) which maps a base density \(p_z(z)\) over the space \(\mathcal{Z} = \mathbb{R}^n\), typically taken to be a multivariate normal, to \(p_x(x)\) via the change of variables formula

\[
p_x(x) = p_z(f^{-1}(x)) \left| \det J_f(f^{-1}(x)) \right|^{-1},
\]

where \(J_f(z)\) is the Jacobian matrix of \(f\) at the point \(z\). In geometric terms, the probability mass in an infinitesimal volume \(dz\) of \(\mathcal{Z}\) must be preserved in the volume of \(\mathcal{X}\) corresponding to the image \(f(dz)\), and the magnitude of the Jacobian determinant is exactly what accounts for changes in the coordinate volume induced by \(f\). By parameterizing classes of diffeomorphisms \(f_i\), the flow model can be fitted via maximum likelihood on the training data \(\{x_i\} \subset \mathcal{X}\). Altogether, the three following operations must be tractable: sampling with \(f(z)\), inference with \(f^{-1}(x)\), and density estimation with the \(|\det J_f(z)|\) factor. To scale the model, one can compose many such layers \(f = f_k \circ \cdots \circ f_1\), and the \(|\det J_{f_k}(z)|\) 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 invertible layers whose Jacobian determinants are tractable by construction. A significant theme has been to structure flows to have a triangular Jacobian \cite{16,59,37,17}. Kingma and Dhariwal \cite{35} 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 \(O(c)\), the number of input channels. See Papamakarios et al. \cite{45} for a thorough survey of normalizing flows.

2.2 Injective Flows

The requirement that \(f\) be a diffeomorphism fixes the dimensionality of the latent space. In turn, \(p_x(x)\) 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 \cite{14} 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 true data density \(p^*(x)\) entirely. Behrmann et al. \cite{5} point out that analytically 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 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 \textit{smooth embedding}, or an injective function which is
While which case the model’s log-likelihood will be with the entire process depicted in Fig. 1.\footnote{Throughout this work we use “embedding” in the topological sense: a function which describes how a low-dimensional space can sit 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.}

For inference, data points from \(\mathcal{M}\) follow the reverse path through \(g^\dagger\) and \(h^{-1}\) to the latent space \(\mathcal{Z}\) where their densities can be evaluated and combined with the determinant factors in Eq. \footnote{\(\dagger\) denotes a left-inverse function, not necessarily the matrix pseudoinverse.}.

Generating samples from \(\mathbf{Z}\) is,\footnote{\(\mathbf{Z}\) is diffeomorphic to its image. This case can be accommodated with a generalized change of variables formula for densities as follows \cite{22}.} as per \(\mathbf{Z}\) and \(\mathbf{u}\). That is, \(\mathcal{M} = g(\mathcal{U})\) is the range of \(g\). Accordingly, \(g\) has a left-inverse \footnote{\(\mathcal{M}\) and \(\mathcal{U}\) are mapped by a bijective flow \(g\). Suppose \(p_u(\mathbf{u})\) 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 \(g\).} \(g^\dagger : \mathcal{X} \rightarrow \mathcal{U}\) which is smooth on \(\mathcal{M}\) and satisfies \(g^\dagger(g(\mathbf{u})) = \mathbf{u}\) for all \(\mathbf{u} \in \mathcal{U}\). Suppose \(p_u(\mathbf{u})\) 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 \(g\).

The quantity that accounts for changes in the coordinate volume at each point \(\mathbf{u}\) is \(\sqrt{\det [J_g^T(\mathbf{u})J_g(\mathbf{u})]}\), where the Jacobian \(J_g\) is now a \(n \times m\) matrix \footnote{\(\mathcal{M}\) must be injective to allow its image \(\mathcal{U}\) to be a manifold. This formulation leads the learned manifold \(\mathcal{M}\) to be diffeomorphic to Euclidean space, which can cause numerical instability when the data’s support differs in topology \cite{14}, but we leave this issue to future work.}. Hence, using the shorthand \(\mathbf{u} = g^\dagger(\mathbf{x})\), the generalized change of variables formula defined for \(\mathbf{x} \in \mathcal{M}\) can be written

\[
p_x(\mathbf{x}) = p_u(\mathbf{u}) \left|\det [J_g^T(\mathbf{u})J_g(\mathbf{u})]\right|^{-\frac{1}{2}}. \tag{2}
\]

While \(g\) describes how the data manifold \(\mathcal{M}\) is embedded in the larger ambient space, the mapping \(g\) 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 \(h : \mathcal{Z} \rightarrow \mathcal{U}\) representing a bijective NF between \(\mathcal{Z}\) and \(\mathcal{U}\) \footnote{If \(\mathcal{M}\) cannot perfectly fit the data, in practice 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. \footnote{Cunningham et al. \cite{13} remedy this by adding} \(\dagger\) to remedy this by adding}. Taking the overall injective transformation \(g \circ h\) and applying the chain rule \(J_{g \circ h} = J_g J_h\) simplifies the determinant in Eq. \footnote{\(\dagger\) denotes a left-inverse function, not necessarily the matrix pseudoinverse.} \eqref{eq:2} since the outer Jacobian \(J_h\) is square,

\[
\det [J_h J_g^T J_g J_h] = (\det J_h)^2 \det [J_g^T J_g]. \tag{3}
\]

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

\[
p_x(\mathbf{x}) = p_z(\mathbf{z}) \left|\det J_h(\mathbf{z})\right|^{-1} \left|\det [J_g^T(\mathbf{u})J_g(\mathbf{u})]\right|^{-\frac{1}{2}}, \tag{4}
\]

with the entire process depicted in Fig. \footnote{\(\dagger\) denotes a left-inverse function, not necessarily the matrix pseudoinverse.}.\[1]
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 work \cite{38, 40, 39} has projected data to the manifold via $g \circ g^\dagger$ prior to computing log-likelihoods and optimized $g$ using the reconstruction loss $\mathbb{E}_{x \sim p_Z} [||x - g(g^\dagger(x))||^2]$. We prove in App. \ref{app:main} that minimizing the reconstruction loss brings the learned manifold into alignment with the data manifold.

When computing log-likelihoods, the determinant term $\log \vert \det J_g \vert$ presents a computational challenge. Kumar et al. \cite{41} maximize it using an approximate lower bound, while Brehmer and Cranmer \cite{39} and Kothari et al. \cite{43} circumvent its computation altogether by only maximizing the other terms in the log-likelihood. In concurrent work, Caterini et al. \cite{44} optimize injective flows using a stochastic estimate of the log-determinant’s gradient. They are also able to optimize $\log \vert \det J_g \vert$ exactly for smaller datasets, but this procedure involves the explicit construction of $J_g$, which would be memory-intensive to scale to larger data such as CelebA. In line with research to build expressive bijective flows where $\det J_f$ is tractable, our work focuses on designing and parameterizing injective flows where $\log \vert \det J_g \vert$ as a whole is efficiently computable. In contrast to past injective flow models, our approach allows for straightforward evaluation and optimization of $\log \vert \det J_g \vert$ in the same way standard NFs do for $\log \vert \det J_f \vert$. 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 $\log \vert \det J_f \vert$ terms by designing layers with triangular Jacobians \cite{46, 47}. For injective flows, the combination $J_g^T J_g$ is symmetric, so it is triangular if and only if it is diagonal. In turn, $J_g^T J_g$ being diagonal is equivalent to $J_g$ having orthogonal columns. While this restriction is feasible for a single layer $g$, it is not composable. If $g_1$ and $g_2$ are both smooth embeddings whose Jacobians have orthogonal columns, it need not follow that $J_{g_1 \circ g_2}$ has orthogonal columns. Additionally, since the Jacobians are not square the determinant in Eq. \ref{eq:det} cannot be factored into a product of individually computable terms as in Eq. \ref{eq:logdet}. To ensure compositability we propose enforcing the slightly stricter criterion that $J_{g_1 \circ g_2}$ be a scalar multiple of the identity. This is precisely the condition that $g$ is a conformal embedding.

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

\[
J_g^T (u) J_g (u) = \lambda^2 (u) I_m ,
\]

where $\lambda : \mathcal{U} \rightarrow \mathbb{R}$ is a smooth non-zero scalar function, the conformal factor \cite{43}. In other words, $J_g$ has orthonormal columns up to a smoothly varying non-zero multiplicative constant. Hence $g$ locally preserves angles.

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

\[
p_Z (x) = p_Z (z) |\det J_h (z)|^{-1} \lambda^{-m} (u) .
\]

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

CEFs provide a new way to coordinate the training dynamics of the model’s manifold and density. It is important to note that not all parameterizations $g$ of the learned manifold are equally suited to density estimation \cite{49}. Prior injective flow models \cite{38, 40, 39} have been trained sequentially by first optimizing $g$ using the reconstruction loss $\mathbb{E}_{x \sim p_Z} [||x - g(g^\dagger(x))||^2]$, then training $h$ for maximum likelihood with $g$ fixed. This runs the risk of initializing the density $p_u (u)$ in a configuration that is challenging for $h$ to learn. Brehmer and Cranmer \cite{39} also alternate training $g$ and $h$, but this does not prevent $g$ from converging to a poor configuration for density estimation. Unlike previous injective flows, CEFs have tractable densities, which allows $g$ and $h$ to be trained jointly by optimizing the loss function

\[
\mathcal{L} = \mathbb{E}_{x \sim p_Z} [-\log p_x (x) + \alpha ||x - g(g^\dagger(x))||^2] .
\]

This mixed loss provides more flexibility in how $g$ is learned, and is unique to our model because it is the first for which $\log p_x (x)$ is tractable.
3.1 Designing Conformal Embedding Flows

Having established the model’s high-level structure and training objective, it remains for us to design conformal embeddings \( g \) which are capable of representing complex data manifolds. For \( g \) to be useful in a CEF we must be able to sample with \( g(u) \), perform inference with \( g^{-1}(x) \), and compute the conformal factor \( \lambda(u) \). In general, there is no unifying way to parameterize the entire family of such conformal embeddings (see App. B for more discussion). As when designing standard bijective flows, we can only identify subfamilies of conformal embeddings which we parameterize and compose to construct an expressive flow. To this end, we work with conformal building blocks \( g_k : \mathcal{U}_i \rightarrow \mathcal{U}_i \) (where \( \mathcal{U}_0 = \mathcal{U} \) and \( \mathcal{U}_k = \mathcal{X} \)), which we compose to produce the full conformal embedding \( g \):

\[
g = g_k \circ \cdots \circ g_1.
\]

In turn, \( g \) is conformal because

\[
J_g^T J_g = (J_{g_k}^T \cdots J_{g_1}^T) (J_{g_k} \cdots J_g) = \lambda_1^2 \cdots \lambda_k^2 I_m.
\]

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. In this special case, \( g \) is called a conformal mapping. Liouville’s theorem 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. B.1 for details on conformal mappings). We created conformal embeddings primarily by composing these layers. Zero-padding is another conformal embedding, with Jacobian \( J_g = (I_m \, 0)^T \), and can be interspersed with conformal mappings to provide changes in dimensionality.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FUNCTIONAL FORM</th>
<th>PARAMS</th>
<th>INVERSE</th>
<th>( \lambda(u) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td>( u \rightarrow u + a )</td>
<td>( a \in \mathbb{R}^d )</td>
<td>( v \rightarrow v - a )</td>
<td>1</td>
</tr>
<tr>
<td>Orthogonal</td>
<td>( u \rightarrow Qu )</td>
<td>( Q \in O(d) )</td>
<td>( v \rightarrow Q^T v )</td>
<td>1</td>
</tr>
<tr>
<td>Scaling</td>
<td>( u \rightarrow \lambda u )</td>
<td>( \lambda \in \mathbb{R} )</td>
<td>( v \rightarrow \lambda^{-1} v )</td>
<td>( \lambda )</td>
</tr>
<tr>
<td>Inversion</td>
<td>( u \rightarrow u/</td>
<td></td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>SCT</td>
<td>( u \rightarrow \gamma + \frac{1}{1 - 2b \cdot u +</td>
<td></td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

Stacking translation, orthogonal transformation, scaling, and inversion layers is sufficient to learn any conformal mapping in principle. However, the inversion operation is numerically unstable, so we replaced it with the special conformal transformation (SCT), a transformation of interest in conformal field theory. It can be understood as an inversion, followed by a translation by \(-b\), 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 implementing conformal mappings was writing trainable orthogonal layers. We parameterized orthogonal transformations in two different ways: by using Householder matrices, which are cheaply parameterizable and easy to train, and by using GeoTorch, the API provided by GeoTorch, which parameterizes the special orthogonal group by taking the matrix exponential of skew-symmetric matrices. GeoTorch also provides trainable non-square matrices with orthonormal columns, which are conformal embeddings (not conformal mappings) and which we incorporate to change the data’s dimensionality.

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, we

---

1We 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.
We considered the two piecewise conformal embeddings defined in Table 2. Due to the success of which are known in advance and can be defined as the image of some fixed was the only example of these that provided performance improvements. To make the embeddings more expressive, the conformality condition on \( U \) with orthonormal columns (in practice, reducing the filter count), we can provide conformal changes in dimension. 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 make the embeddings more expressive, the conformality condition on \( g \) can be relaxed to the point of being conformal almost everywhere. Formally, the latent spaces \( Z \) and \( U \) are redefined as \( Z = \{ z : g \) is conformal at \( h(z) \} \) and \( U = h(Z) \). Then \( g \) remains a conformal embedding on \( U \), and as long as \( \{ x : g \) is nonconformal at \( g^{-1}(x) \} \) 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 [17] and can be justified by generalizing the change of variables formula [38].

### Table 2: Piecewise Conformal Embeddings

<table>
<thead>
<tr>
<th>Type</th>
<th>Functional Form</th>
<th>Params</th>
<th>Left Inverse</th>
<th>( \lambda(u) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conformal ReLu</td>
<td>( u \mapsto \text{ReLU} \left[ \begin{array}{l} Q_1 u \ -Q_2 u \end{array} \right] )</td>
<td>( Q \in O(d) )</td>
<td>( \begin{array}{l} v_1 \ v_2 \end{array} \mapsto Q^T (v_1 - v_2) )</td>
<td>1</td>
</tr>
<tr>
<td>Conditional Orthogonal</td>
<td>( u \mapsto \left{ \begin{array}{l} Q_1 u \quad \text{if } |u| &lt; 1 \ Q_2 u \quad \text{if } |u| \geq 1 \end{array} \right. )</td>
<td>( Q_1, Q_2 \in O(d) )</td>
<td>( \begin{array}{l} v \mapsto \left{ \begin{array}{l} Q_1^T u \quad \text{if } |v| &lt; 1 \ Q_2^T u \quad \text{if } |v| \geq 1 \end{array} \right. )</td>
<td>1</td>
</tr>
</tbody>
</table>

We considered the two piecewise conformal embeddings defined in Table 2. Due to the success of ReLU in standard deep neural networks [50], we try a ReLU-like layer that is piecewise conformal. conformal ReLU is based on the injective ReLU proposed by Kothari et al. [39]. We believe it to be of general interest as a dimension-changing conformal nonlinearity, but it provided no performance improvements in experiments.

More useful was the conditional orthogonal transformation, which takes advantage of the norm-preservation of orthogonal transformations to create an invertible layer. Despite being discontinuous, it provided a substantial boost in reconstruction ability on image data. The idea behind this transformation can be extended to the other parameterized mappings in Table 1. For each type of mapping we can identify hypersurfaces in \( \mathbb{R}^n \) such that each hypersurface is mapped back to itself; i.e., each hypersurface is an orbit of its points under the mapping. Applying the same type of conformal mapping piecewise on either side remains an invertible operation as long as trajectories do not cross the hypersurface, and the result is conformal almost everywhere. The conditional orthogonal layer was the only example of these that provided performance improvements.

### 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 \( \phi : \mathcal{U} \rightarrow \mathcal{X} \), where \( \mathcal{U} \subseteq \mathbb{R}^n \) [22, 48, 24]. In particular, Rezende et al. [60] 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 Jacobian determinant factor may be computable in closed form. Our work replaces \( \phi \) with a trainable network \( g \), but the log-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.

---

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. Moreover, by modifying these matrices to be non-square with orthonormal columns (in practice, reducing the filter count), we can provide conformal changes in dimension. 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.
When the distribution is manifold-supported, it will assign zero density to off-manifold points. This has been addressed by adding an off-manifold noise term [12, 13] or by projecting the data onto the manifold and training it with a reconstruction loss [6, 39, 41]. 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. [41] use a series of lower bounds to train the log-determinant, while Brehmer and Cranmer [6] and Kothari et al. [39] separate the flow into two components and train only the low-dimensional component. Caterini et al. [9] maximize log-likelihood directly by either constructing the embedding’s Jacobian explicitly or using a stochastic approximation of the log-determinant’s gradient, but both approaches remain computationally expensive for high-dimensional data. Our approach is the first injective model to provide a learnable manifold with exact and efficient log-determinant computation.

**Conformal networks.** Numerous past works have imposed approximate conformality or its special cases as a regularizer [4, 31, 56, 57, 70], but it has been less common to enforce conformality strictly. To maintain orthogonal weights, one must optimize along the Stiefel manifold of orthogonal matrices. Past work to achieve this has either trained with Riemannian gradient descent or directly parameterized subsets of orthogonal matrices. Riemannian gradient descent algorithms typically require a singular value or QR decomposition at each training step [26, 30, 53]. We found that orthogonal matrices trained more quickly when directly parameterized. In particular, Lezcano-Casado and Martínez-Rubio [45] and Lezcano-Casado [44] parameterize orthogonal matrices as the matrix exponential of a skew-symmetric matrix, and Tomczak and Welling [67] use Householder matrices. We used a mix of both.

## 5 Experiments

To implement CEFs, we worked off of the nflows github repo [20], which is derived from the code of Durkan et al. [19]. Our code is available at https://github.com/layer6ai-labs/CEF. Full model and training details are provided in App. C, while additional reconstructions and generated images are presented in App. D.

### 5.1 Spherical Data

To demonstrate 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. C. The distribution is visualized in Fig. 2 along with the training dataset of $10^3$ sampled points.

We trained the two components of the CEF jointly, using 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. It shows good fidelity to the true manifold and density.

![Figure 2](image)

Figure 2: (a) A density $p^*_x(x)$ with support on the sphere, and (b) $10^3$ samples comprising the training dataset $\{x_1\}$. (c) The density learned by a CEF, and (d) $10^3$ generated samples.

### 5.2 Image Data

We now evaluate manifold flows on image data. Our aim is to show that, although they represent a strict architectural subset of mainstream injective flows, CEFs remain competitive in generative performance [6, 59]. In doing so, this work is the first to include end-to-end maximum likelihood
training with an injective flow on image data. Three approaches were evaluated on each dataset: a jointly trained CEF, a sequentially trained CEF, and for a baseline a sequentially trained injective flow, as in Brehmer and Cranmer [6], labelled manifold flow (MF).

Injective models cannot be compared on the basis of log-likelihood, since each model may have a different manifold support. Instead, we evaluate generative performance in terms of fidelity and diversity [61]. The FID score [28] is a single metric which combines these factors, whereas density and coverage [49] measure them separately. For FID, lower is better, while for density and coverage, higher is better. We use the PyTorch-FID package [62] and the implementation of density and coverage from Naeem et al. [49].

**Synthetic image manifolds.** Before graduating to natural high-dimensional data, we test CEFs on a high-dimensional synthetic data manifold whose properties are better understood. We generate data using a GAN pretrained on CIFAR-10 [40] by sampling from a selected number of latent space dimensions (64 and 512) with others held fixed. Specifically, we sample a single class from the class-conditional StyleGAN2-ADA provided by Karras et al. [33]. This setup reflects our model design in that (1) the true latent dimension is known and (2) since a single class is used, the resulting manifold is more likely to be connected. On the other hand, the GAN may not be completely injective, so its support may not technically be a manifold. Results are shown in Table 3, Figs. 3 and 4, and App. D.

### Table 3: Synthetic CIFAR-10 Ship Manifolds

<table>
<thead>
<tr>
<th>Model</th>
<th>64 Dimensions</th>
<th>512 Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RECON</td>
<td>FID</td>
</tr>
<tr>
<td>J-CEF</td>
<td>0.000695</td>
<td>36.5</td>
</tr>
<tr>
<td>S-CEF</td>
<td>0.000717</td>
<td>35.3</td>
</tr>
<tr>
<td>S-MF</td>
<td>0.000469</td>
<td>28.7</td>
</tr>
</tbody>
</table>

Figure 3: Synthetic 512-dimensional Ship Manifold Reconstructions. From top to bottom: groundtruth samples, joint CEF, sequential CEF, and sequential MF.

Figure 4: Uncurated Synthetic 512-dimensional Ship Manifold Samples. From top to bottom: joint CEF, sequential CEF, and sequential MF.
Table 4: Natural Image Data

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST Recon</th>
<th>MNIST FID</th>
<th>MNIST Density</th>
<th>MNIST Cov</th>
<th>CelebA Recon</th>
<th>CelebA FID</th>
<th>CelebA Density</th>
<th>CelebA Cov</th>
</tr>
</thead>
<tbody>
<tr>
<td>J-CEF</td>
<td>0.003222</td>
<td>38.5</td>
<td>0.0725</td>
<td>0.1796</td>
<td>0.001016</td>
<td>118</td>
<td>0.05581</td>
<td>0.00872</td>
</tr>
<tr>
<td>S-CEF</td>
<td>0.003315</td>
<td>37.9</td>
<td>0.0763</td>
<td>0.1800</td>
<td>0.001019</td>
<td>171</td>
<td>0.00922</td>
<td>0.00356</td>
</tr>
<tr>
<td>S-MF</td>
<td>0.000491</td>
<td>16.1</td>
<td>0.5003</td>
<td>0.7126</td>
<td>0.000547</td>
<td>142</td>
<td>0.02425</td>
<td>0.00576</td>
</tr>
</tbody>
</table>

Figure 5: MNIST Reconstructions. From top to bottom: groundtruth samples, joint CEF, sequential CEF, and sequential MF.

All models achieve comparable reconstruction losses with very minor visible artifacts, showing that conformal embeddings can learn complex manifolds with similar efficacy to state-of-the-art flows, despite their restricted architecture. The manifold flow achieves better generative performance based on our metrics and visual clarity. Between the CEFs, joint training allows the learned manifold to adapt better, but this did not translate directly to better generative performance.

Natural image data. We scale CEFs to natural data by training on the MNIST \[42\] and CelebA \[46\] datasets, for which a low-dimensional manifold structure is postulated but unknown. Results are given in Table 4, Figs. 5 and 6, and App. D.

As expected, since the MF’s embedding is more flexible, it achieves smaller reconstruction losses than the CEFs. On MNIST, this is visible as faint blurriness in the CEF reconstructions in Fig. 5, and it translates to better sample quality for the MF as per the metrics in Table 4. Interestingly however, the jointly-trained CEF obtains substantially better sample quality on CelebA, both visually (Fig. 6) and by every metric. We posit this is due to the phenomenon observed by Caterini et al. \[9\] in concurrent work: for complex distributions, the learned manifold parameterization has significant influence on the difficulty of the density estimation task. Only the joint training approach, which maximizes likelihoods end-to-end, can train the manifold parameterization to an optimal starting point for density estimation, while sequential training optimizes the manifold solely on the basis of reconstruction loss. CelebA is the highest-dimensional dataset tested here, and its distribution is presumably quite complex, so one can reasonably expect joint training to provide better results. On the other hand, the sequentially trained CEF’s performance suffers from the lack of both joint training and the expressivity afforded by the more general MF architecture.

6 Limitations and Future Directions

Expressivity. Just as standard flows trade expressivity for tractable likelihoods, so must injective flows. Our conformal embeddings in particular are less expressive than state-of-the-art flow models; they had higher reconstruction loss than the neural spline flow-based embeddings we tested. The conformal embeddings we designed were limited in that they mostly derive from dimension-preserving conformal mappings, which is a naturally restrictive class by Liouville’s theorem \[27\]. Just as early work on NFs \[16, 59\] introduced limited classes of parameterizable bijections, which were later improved substantially (e.g. \[19, 35\]), our work introduces several classes of parameterizable conformal embeddings. We expect that future work will uncover more expressive conformal embeddings.
Figure 6: Uncurated CelebA Samples. From top to bottom: joint CEF, sequential CEF, and sequential MF.

**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. The difference in supports between two manifold models also makes their likelihoods incomparable. Cunningham et al. have made progress in this direction by convolving the manifold-supported distribution with noise, but this makes inference 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.

**Broader impact.** As deep generative models become more advanced, researchers should carefully consider some accompanying ethical concerns. Large-scale, natural image datasets carry social biases which are likely to be codified in turn by the models trained on them. For instance, CelebA does not accurately represent the real-world distribution of human traits, and models trained on CelebA should be vetted for fairness before being deployed to make decisions that can adversely affect people. Deep generative modelling also lends itself to malicious practices such as disinformation and impersonation using deepfakes.

Our work seeks to endow normalizing flows with more realistic assumptions about the data they model. While such improvements may invite malicious downstream applications, they also encode a better understanding of the data, which makes the model more interpretable and thus more transparent. We hope that a better understanding of deep generative models will synergize with current lines of research aimed at applying them for fair and explainable real-world use.

7 Conclusion

This paper introduced Conformal Embedding Flows for modelling probability distributions on low-dimensional manifolds while maintaining tractable densities. We showed that conformal embeddings naturally match the framework of normalizing flows by providing efficient sampling, inference, and density estimation, and they are composable so that they can be scaled to depth. Furthermore, it appears conformality is a minimal restriction in that any looser condition will sacrifice one or more of these properties. As we have reviewed, previous instantiations of injective flows have not maintained 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 research has made progress towards VAE- or GAN-level performance. The CEF paradigm is a way to match these strides while maintaining the theoretical strengths of NFs.
Acknowledgments and Disclosure of Funding

We thank Gabriel Loaiza-Ganem and Anthony Caterini for their valuable discussions and advice. We also thank Parsa Torabian for sharing his experience with orthogonal weights and Maksims Volkovs for his helpful feedback.

The authors declare no competing interests or third-party funding sources.

References


