Draft version October 12, 2023
Typeset using LaTeX **twocolumn** style in AASTeX631

Statistical Modeling of Galaxy Catalogues with Normalizing Flows

John Franklin Crenshaw,[1,2] J. Bryce Kalmbach,[1] Alexander Gagliano,[3,4,5,6] Ziang Yan,[7] Andrew J. Connolly,[1] Alex I. Malz,[8] Samuel J. Schmidt,[9]

The LSST Dark Energy Science Collaboration

[1] *DIRAC Institute and the Department of Astronomy, University of Washington, Seattle, WA 98195, USA*
[2] *Department of Physics, University of Washington, Seattle, WA 98195, USA*
[3] *Department of Astronomy, University of Illinois at Urbana-Champaign, 1002 W. Green St., IL 61801, USA*
[4] *National Center for Supercomputing Applications, Urbana, IL, 61801, USA*
[5] *Center for AstroPhysical Surveys, Urbana, IL, 61801, USA*
[6] *National Science Foundation Graduate Research Fellow*
[7] *Ruhr University Bochum, Faculty of Physics and Astronomy, Astronomical Institute (AIRUB), German Centre for Cosmological Lensing, 44780 Bochum, Germany*
[8] *McWilliams Center for Cosmology, Department of Physics, Carnegie Mellon University*
[9] *Department of Physics and Astronomy, University of California, One Shields Avenue, Davis, CA 95616, USA*

ABSTRACT

Normalizing flows are powerful tools for learning high-dimensional probability distributions from samples thereof, and have many applications in astronomy, including posterior estimation and forward modeling. We introduce PZFlow, a Python package for statistical modeling of tabular data using normalizing flows. We use PZFlow to model photometric galaxy catalogs including redshifts, photometry, size, and shape, and generate a synthetic catalog. In this catalog, each galaxy has a *true* redshift posterior. We discuss the importance of these true posteriors for the comprehensive evaluation of redshift posteriors produced by photometric redshift (photo-z) estimators. We also demonstrate the use of an ensemble of normalizing flows for density estimation, applied to photo-z estimation. While we focus on photo-z estimation and validation, we emphasize that these methods are applicable to any galaxy properties, and any other tabular data.

1. INTRODUCTION

Astronomical data represent realizations of complex probability distributions. A common goal in research is to infer the underlying distribution from a limited set of noisy data, and use this distribution to estimate posterior distributions over galaxy and population parameters. An important example is photometric redshift (photo-z) estimation, where galaxy redshift posteriors are estimated from galaxy photometry, using a model informed by a training set of spectroscopic galaxy data (Newman & Gruen 2022). These redshift posteriors are then used to estimate posterior distributions for cosmological parameters (The LSST Dark Energy Science Collaboration et al. 2018).

Corresponding author: John Franklin Crenshaw
jfc20@uw.edu

Knowledge of the probability distributions underlying our data is also valuable for simulation and forward modeling astronomical data sets. Simulating data sets with realistic statistical properties enables methodological development and the calibration of systematic uncertainties. Forward modeling also facilitates data augmentation (e.g. Lokken et al. 2022) and the creation of large data challenges (e.g. Kessler et al. 2019; LSST Dark Energy Science Collaboration et al. 2021; Korytov et al. 2019), which are becoming more prevalent in the big data era of astronomy.

Inferring the underlying probability distribution, or likelihood, from a high-dimensional data set is a difficult problem. Many machine learning tools have been developed for this task, including Generative Adversarial Networks (GANs; Goodfellow et al. 2014) and Variational Autoencoders (VAEs; Kingma & Welling 2014). Both are neural networks that excel at forward modeling complex data sets, but neither allow exact likeli-

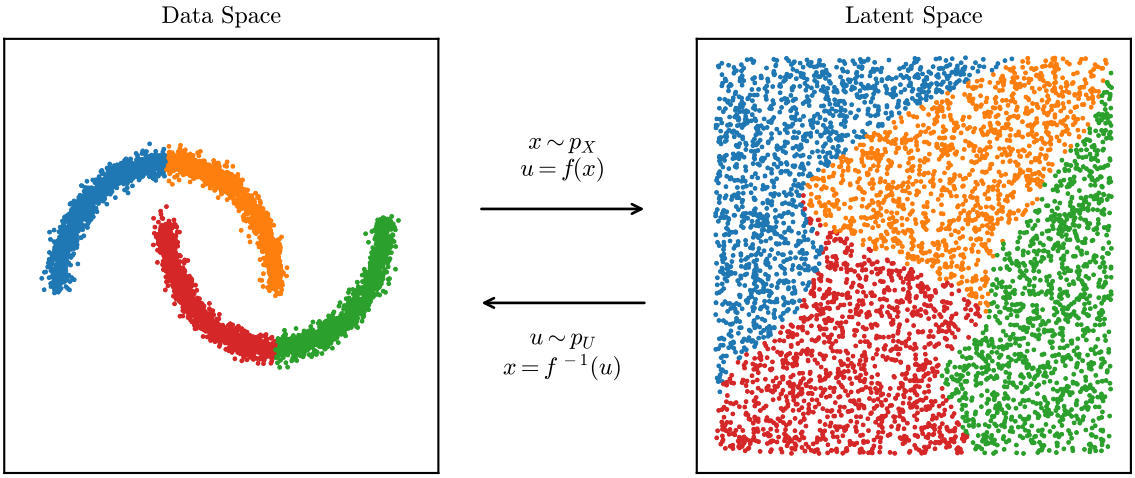

Figure 1. A normalizing flow demonstrated on the two moons data set from scikit-learn. The two moons data on the left is mapped onto a two dimensional uniform distribution by the bijection f. The data are colored by quadrant to visualize their image in the latent space. You can sample the data distribution by sampling from the uniform distribution, and using f^{-1} to map the samples back to the data space.

hood calculation for the simulated data. This means that questions such as "under my model, what is the posterior distribution for redshift given the simulated photometry" have only approximate answers.

Normalizing flows, on the other hand, are a deep learning tool that excel at forward modeling, while also allowing exact, analytic likelihood calculation. In other words, they allow you to analytically calculate likelihoods and posteriors with respect to the distribution from which simulated data is drawn. Normalizing flows operate by learning an invertible transformation of the data distribution into a simpler, tractable distribution, known as the latent distribution. A common choice for latent distribution is a normal distribution, hence the name *normalizing* flow. This allows sampling and likelihood calculation to be performed within the context of the simple latent distribution, with the normalizing flow acting as a translator between the latent samples and likelihoods, and their corresponding values in the context of the complicated data distribution.

Normalizing flows have gained popularity as tools for efficient and flexible sampling for parameter inference (e.g., Dai & Seljak 2022; Dacunha et al. 2022; Hassan et al. 2022). In this paper, we focus on photo-z's and forward modeling photometric galaxy catalogs. Since normalizing flows allow exact, analytic probability calculation, the properties of objects in these catalogs have *true* posteriors. This means that questions such as "under my model, what is the posterior distribution for redshift given the simulated photometry" have exact answers. Catalogs with true posteriors are useful for the testing and validation of analysis pipelines that estimate posteriors, such as the photo-z estimators used in much of astrophysics and cosmology. Previous evaluations of these estimators have focused on comparing point estimates to true values (e.g., Hildebrandt et al. 2010; Sánchez et al. 2014; Graham et al. 2018), or evaluating ensembles of posteriors (Schmidt & Malz et al. 2020). Normalizing flow catalogs with true posteriors open up a new, more comprehensive avenue for evaluation of these estimators by enabling direct posterior-to-posterior comparison.

To facilitate the statistical modeling of galaxy catalogs and other astronomical data sets, we have developed PZFlow, a normalizing flow package for Python. With relatively little tuning required by the user, PZFlow can provide a generative model for any tabular data, including continuous and discrete variables, and variables with Euclidean or periodic topology. In addition to generative modeling, PZFlow can calculate posteriors over any columns in your data set, and can convolve errors and marginalize over missing columns while training the model or calculating posteriors for samples.

In this paper, we provide the background on normalizing flows (Section 2) required to understand PZFlow (Section 3). We then use PZFlow to simulate a galaxy catalog (Section 4), where each object has photometry, size, ellipticity, redshift, and a true redshift posterior. We also demonstrate using PZFlow as a density estimator, via the example of photo-z estimation (Section 5). We conclude in Section 6.

2. NORMALIZING FLOWS

Normalizing flows model complex, high-dimensional probability distributions by learning a mapping from the

data distribution to a tractable latent distribution[1]. Often the latent distribution is a standard Normal distribution, and so the mapping "normalizes" the data, hence the name "normalizing flow". This mapping allows us to sample and evaluate densities using the latent distribution, rather than the unknown data distribution.

Assume we have a differentiable function f that maps samples x from the data distribution p_X onto samples u from the latent distribution p_U. Using the change of variables formula, we can evaluate the probability density of the data:

$$p_X(x) = p_U(u = f(x)) \, |\det \nabla f(x)|, \tag{1}$$

where $\nabla f(x)$ is the Jacobian of f evaluated at x. In words, computing the density $p_X(x)$ is accomplished by mapping x to the latent distribution, calculating its density there, and multiplying by the associated Jacobian determinant, which accounts for how the function f distorts volume elements of the space.

If we further assume that f is invertible, we can sample from the data distribution by applying f^{-1} to samples from the latent distribution[2]:

$$x = f^{-1}(u) \quad \text{where} \quad u \sim p_U. \tag{2}$$

Figure 1 shows an example of a normalizing flow that transforms the scikit-learn (Pedregosa et al. 2011) two moons distribution into a uniform distribution. The data points are colored by quadrant to visualize their image under f.

The following sections discuss how to build a normalizing flow to model data with various features. Section 2.1 discusses the bijection f and introduces the building blocks from which our bijections will be built; Section 2.2 discusses how to choose an appropriate latent distribution for your data; Section 2.3 describes how to build a flow that models a conditional distribution; Section 2.4 explains how to model data with periodic topology; finally Section 2.5 explains how to model data with discrete variables.

2.1. *Designing a bijection*

A bijection is an invertible map between two sets. In a normalizing flow, the bijection maps the data distribution onto the latent distribution for likelihood calculation, and the inverse of the bijection maps samples from the latent distribution back to the data distribution. The bijection of a normalizing flow must be powerful enough to model complex relationships in data, while remaining invertible and simultaneously possessing an efficiently computable Jacobian determinant. This latter constraint is the primary difficulty in designing a normalizing flow. The most popular strategy for achieving these requirements is to exploit the fact that a composition of bijections is also bijective. By chaining together multiple less-expressive bijections whose Jacobians are efficiently computable, a composite bijections can be constructed that meets our requirements:

$$f = \cdots \circ f_3 \circ f_2 \circ f_1. \tag{3}$$

The overall Jacobian determinant can be efficiently calculated using the chain rule.

There is an extensive literature on constructing these sub-bijections (see Kobyzev et al. 2020 for a review). Some bijections are specialized to be particularly efficient at either density estimation or sampling, but for many science cases, we wish to do both. For this reason, we will focus on Rational-Quadratic Rolling Spline Couplings (RQ-RSCs), bijections which achieve state-of-the-art performance, while being efficient with both tasks (Durkan et al. 2019).

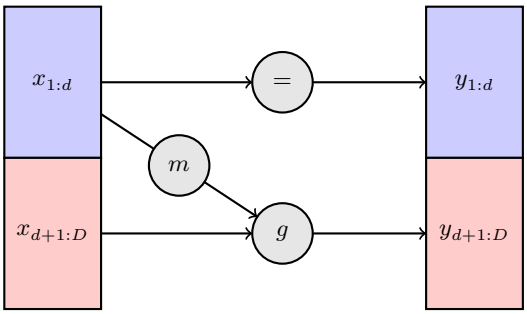

Figure 2. Diagram of a coupling layer. The first partition, $x_{1:d}$, is passed through the layer unchanged. The second partition, $x_{d+1:D}$, is transformed by the coupling law g, which is parameterized by the coupling function m applied to the first partition.

2.1.1. *Rational-Quadratic Rolling Spline Couplings*

RQ-RSCs are bijections that are composed of coupling layers (Dinh et al. 2015, 2017). A coupling layer partitions the data, $x \in \mathbb{R}^D$, into two sets, $x_{1:d}$ and $x_{d+1:D}$. The first set is then used to transform the second set:

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= g(x_{d+1:D}; m(x_{1:d})), \end{aligned} \tag{4}$$

where $g : \mathbb{R}^{D-d} \times \mathbb{R}^d \to \mathbb{R}^{D-d}$ is an invertible *coupling law*, and m is a *coupling function* defined on $\mathbb{R}^d$. This is illustrated in Figure 2. The advantage of this structure

[1] Some of the machine learning literature defines the mapping in the opposite direction.

[2] Here, $\sim$ means "is drawn from."

is that the Jacobian is triangular,

$$\frac{\partial y}{\partial x} = \begin{pmatrix} I_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}} & \frac{\partial y_{d+1:D}}{\partial x_{d+1:D}} \end{pmatrix}, \tag{5}$$

where I_d is the $d \times d$ identity matrix. In particular, the Jacobian determinant is

$$\det \frac{\partial y}{\partial x} = \det \frac{\partial y_{d+1:D}}{\partial x_{d+1:D}}. \tag{6}$$

Furthermore, the inverse can be calculated as

$$\begin{aligned} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= g^{-1}(y_{d+1:D}; m(x_{1:d})), \end{aligned} \tag{7}$$

Notice that neither inverting a coupling layer g, nor calculating the Jacobian determinant, requires inverting or taking derivatives of the coupling function m, which can thus be arbitrarily complex.

The obvious limitation of a coupling layer is that only a subset of the data dimensions are transformed. This is overcome by stacking multiple coupling layers in succession, and switching which variables belong to which partition. In practice, this is achieved by interspersing coupling layers with bijections that shuffle the dimensions of x. These shuffling bijections are trivially inverted and have a Jacobian determinant of one.

In a general coupling layer g, there are a variety of coupling laws m one can use. RQ-RSC's use Rational-Quadratic Neural Spline Coupling (Durkan et al. 2019). As the name suggests, the coupling law g is a set of rational-quadratic splines. In particular, $g_i : [-B, B] \rightarrow [-B, B]$ for each dimension i of $x_{d+1:D}$, where g_i is a piecewise combination of K segments, and each segment is a rational-quadratic function. The positions and derivatives of the knots that parameterize the splines are calculated using the coupling function m, which is a dense neural network applied to $x_{1:d}$.

The result is a bijection that achieves state-of-the-art performance and efficiency for forward modeling and density estimation (Kobyzev et al. 2020), and are flexible enough to model complex distributions with multiple discontinuities and hundreds of modes. In addition, they are easily adaptable for flows with periodic topology (Section 2.4). For more details, see Durkan et al. (2019).

In this work, we stack Rational-Quadratic Neural Spline Couplings, with Rolling Layers between each – a configuration we name Rational-Quadratic Rolling Spline Couplings (RQ-RSCs). Rolling Layers shift the dimensions of x by one place:

$$\text{Roll} : [x_1, \ldots, x_{D-1}, x_D] \rightarrow [x_D, x_1, \ldots, x_{D-1}]. \tag{8}$$

By constructing a stack with D coupling layers, RQ-RSCs individually transform each of the D dimensions of x as a function of the other $D - 1$ dimensions. In the limit of high spline resolution (i.e. $K \rightarrow \infty$), RQ-RSCs can model any differentiable, monotonic function on $[-B, B]^D$ and can thus model arbitrarily complex distributions in this region. In practice, we find very good performance for diverse data sets with $K \approx 16$.

Note you can specify a different value of K for each of the D spline layers in order to individually control the resolution of each dimension. Lowering K typically results in a smoother distribution, while increasing K increases the complexity the normalizing flow can capture, while also increasing computational and memory cost.

2.1.2. *Data processing bijections*

While RQ-RSCs perform the heavy lifting of mapping the data distribution p_X onto the latent distribution p_U, it is also convenient to define other bijections that perform useful operations such as pre- and post-processing. We name these *data processing bijections*.

For example, RQ-RSCs (and the RQ-NSCs on which they are based) are defined on the domain [-B, B], and thus will not transform samples outside this range. It is therefore useful to define a *Shift Bounds* bijection, which shifts the original range of each dimension to match the domain of the splines. Note this shift must be set at training time, with the assumption that future test data will lie within the same bounds[3]. You can choose a range wider than that covered by the training set if you wish to allow the flow to sample outside the range of the training set

For an example of building an application-specific data processing bijection, see the *Color Transform* bijection defined in Section 4.1, which maps galaxy magnitudes to galaxy colors. See section 2.5 for data processing bijections that enable modeling of discrete data.

Instead of using these data processing bijections, you can of course manually pre-process the data before evaluating densities and post-process samples drawn from the normalizing flow. However, by building pre- and post-processing directly into the bijection, you remove these extra steps from the workflow. This reduces the complexity of working with the normalizing flow and ensures that the flow always "remembers" how to correctly perform these pre- and post-processing steps.

[3] While this sounds quite restrictive, neural networks are typically pretty bad at extrapolating beyond the bounds of the training set anyway.

2.2. *Choosing a latent distribution*

In principle, with a sufficiently expressive bijection, the choice of latent distribution does not matter as long as it is a distribution in which you can easily sample and calculate densities. However, in practice, bijections are limited in expressiveness, i.e. they cannot necessarily transform any arbitrary data distribution into any arbitrary latent distribution.

For example, the splines of RQ-RSCs only transform samples in the range [-B, B]. Sampling from a latent distribution with support outside this range will therefore result in strange outliers and incorrect boundary conditions. One can apply a transformation to the latent samples before they are fed into the RQ-RSC to ensure that they lie within the support of the splines, but it is simpler to use a compact latent distribution whose support matches that of the spline layers. A simple choice would be the uniform distribution over [-B, B].

Additionally, as no bijection is perfect, the structure of the latent distribution will not be completely erased in the translation from latent to data distribution. Thus, the latent distribution can be viewed as a prior or inductive bias on samples from the data distribution (Jaini et al. 2020). It is therefore advantageous to select a latent distribution whose features match some of the structure in the data.

A latent distribution that can achieve both desiderata is the Beta distribution, i.e. $u \sim \text{Beta}(\alpha, \beta)$, where $\alpha, \beta > 0$ are learnable parameters[4]. This distribution is compact, and by varying α and β this distribution can take on a wide variety of shapes with different means, skews, and kurtoses, allowing the inductive bias of the prior to adapt to structure in the data during training. However, as RQ-RSCs are defined on the domain $[-B, B]$, it is more convenient to use a modified Beta distribution, which we name the *Centered Beta distribution*:

$$\text{CentBeta}(u|\alpha, \beta, B) = 2B \left(\text{Beta}(u|\alpha, \beta) - \frac{1}{2} \right). \quad (9)$$

In general, as long as sampling and density evaluation are tractable, one can use any parameterization of the latent distribution that matches some desired structure in the data and learn the distribution parameters during training. We give this generalization the name *latent-adaptive flows* (LAFs; inspired by the Tail Adaptive Flows of Jaini et al. 2020). Our experiments indicate that learnable latent distributions can improve training loss, but require more care in training.

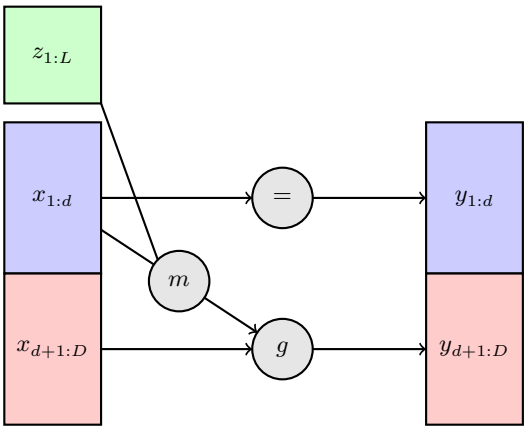

Figure 3. Diagram of a *conditional* coupling layer. The first partition, $x_{1:d}$, is passed through the layer unchanged. The second partition, $x_{d+1:D}$, is transformed by the coupling law g, which is parameterized by the coupling function m applied to the first partition *and* the conditional variables $z_{1:L}$. The conditional variables are *never* altered by the flow.

Note that while we discussed univariate distributions above, these considerations generalize easily to multiple dimensions. Each of these distributions have multivariate generalizations that can be used when modeling higher-dimensional data. The full multivariate latent distribution can also be assembled by taking the product of multiple univariate distributions[5]. This may even be desired if different dimensions of the data have different structure that you wish to encode in the latent distribution.

2.3. *Conditional flows*

The bijections and latent distributions discussed above can be easily adapted to directly learn conditional probability distributions: you only need to make the replacement $f(x) \to f(x; z)$, where z is a vector of conditions (Winkler et al. 2019). This is illustrated in Figure 3, which is a modification of Figure 2 to include the input of conditional variables to the coupling function m. In practice, since m is usually a neural network, this amounts to just appending the conditions z to the inputs of the neural network.

While $p(x|z)$ is technically encoded within $p(x, z)$, which can be learned with a regular normalizing flow, directly modeling $p(x|z)$ with a conditional flow has a few benefits. Training is typically faster, since the latent distribution has a smaller number of dimensions. You can also draw samples of x at fixed values of the conditions z, and you can calculate $p(x|z)$ without having to

[4] In practice, it is easier to learn $\log \alpha$ and $\log \beta$ to ensure that $\alpha, \beta > 0$.

[5] Note that while the latent variables will be independent, the data variables will still have correlations imprinted by the bijections.

numerically calculate and divide by $p(z)$, which can be computationally expensive.

2.4. *Flows with periodic topology*

The flows we have considered so far model data that live in $\mathbb{R}^n$. This assumption is insufficient for modeling variables from spaces with non-Euclidean topology, e.g. positions on the sky. While progress has been made on building flows for general topologies (e.g. Gemici et al. 2016 and Falorsi et al. 2019), we will focus on building flows on the sphere, S^2, as this is the case most relevant in astronomy. We will see that by carefully choosing the latent space, we can construct flows with periodic topology with minimal additional effort (Rezende et al. 2020).

Positions on the sphere are specified by two angles[6], θ and ϕ, the latter of which is periodic. By mapping θ to $\cos\theta$, we map the sphere to a cylinder[7]: $S^2 \to [-1, 1] \times S^1$ (i.e. the Cartesian product of an interval and a circle). In other words, we can transform $\cos\theta$ with a Euclidean flow, as long as we ensure that the flow bounds samples to the range $[-1, 1]$. However, the S^1 piece, ϕ, has a periodic topology and must be handled more carefully.

First, we will address transformations of $\cos\theta$. The only constraint we must impose is that samples of $\cos\theta$ must lie in the range $[-1, 1]$. Fortunately, RQ-RSCs are bounded, mapping a range in u to the same range in x. Thus, if we pick a latent distribution with compact support in $[-1, 1]$, samples of $\cos\theta$ are guaranteed to lie in the same range, as long as we set the range of the RQ-RSC $B = 1$.

Next we will address transformations of ϕ. For f to be a differentiable bijection on the circle, S^1, it is sufficient that f obey the following constraints:

$$f(0) = 0 \tag{10}$$

$$f(2\pi) = 2\pi \tag{11}$$

$$\nabla f(0) = \nabla f(2\pi) \tag{12}$$

$$\nabla f(\phi) > 0. \tag{13}$$

The first two constraints ensure continuity of f by designating $\phi = 0$ as a fixed point, and the third constraint ensures continuity of ∇f at that fixed point. While the designation of $\phi = 0$ as a fixed point is an unnecessary restriction on f, any differentiable bijection on the circle has at least one fixed point up to a phase change, and so this restriction does not actually restrict the expressiveness of f. The fourth restriction ensures monotonicity, which guarantees invertibility.

If we make the phase change $\phi \to \phi - \pi$ so that our angles $\phi \in [-\pi, \pi]$, a RQ-NSC with $B = \pi$ automatically fulfills all four constraints. In fact, regular RQ-NSC's impose the further condition

$$\nabla f(-\pi) = \nabla f(\pi) = 1 \tag{14}$$

to match an identity transform for inputs outside of the range $[-\pi, \pi]$. By choosing a latent distribution with compact support in the range $[-\pi, \pi]$, we ensure that no samples will lie outside the range of the splines, and so we can relax the boundary condition of Equation 14 in favor of the boundary condition in Equation 12. Spline transforms with this relaxed boundary condition are named *Circular Splines* by Rezende et al. (2020).

The circular spline construction above is easily generalized to n-spheres and n-tori: $S^n \to [-1, 1]^{n-1} \times S^1$ and $T^n \to (S^1)^n$ (see Rezende et al. 2020 for more details). We can model the joint distribution of periodic and non-periodic variables with RQ-RSCs simply by choosing appropriate bounds B for each dimension, and by swapping boundary condition 14 for condition 12 for any periodic dimensions.

2.5. *Modeling discrete variables*

In addition to the continuous variables described above, normalizing flows can also be used to model discrete variables. This can be achieved by "dequantizing" the discrete dimensions of the data, which can then be mapped onto continuous latent distributions using regular continuous bijections. Dequantization consists of adding some kind of continuous noise to the discrete dimensions, transforming them into continuous dimensions. When sampling from the flow, you simply do the opposite, and "quantize" the discrete dimensions after applying all of the regular bijections, mapping the noisy, continuous variables onto their discrete counterparts.

A common method for dequantization is uniform dequantization, in which random uniform noise in the range (0, 1) is added to the discrete dimensions. The corresponding quantization applied while sampling from the flow consists of applying the floor function to the dequantized dimensions, mapping these samples onto the nearest integer less than the sampled value. More sophisticated dequantization schemes use variational inference or even another normalizing flow to determine the noise distributions, which improves results by smoothing the discontinuities between neighboring discrete values. See Ho et al. (2019) Hoogeboom et al. (2020) for more details.

[6] We use the convention where θ and ϕ are the zenith and azimuthal angles, respectively.

[7] This map can be explicitly constructed via an embedding in $\mathbb{R}^3$. Technically, the map is not defined for $\theta \in \{0, \pi\}$, however as this set has zero measure, it can be safely ignored.

While the dequantizers are not technically bijections, they can be treated as data processing bijections and be chained together with the other bijections in your normalizing flow.

3. PZFLOW

PZFlow is a Python package for building normalizing flows, with a focus on easy high-performance modeling of high-dimensional tabular data. Data is handled in Pandas DataFrames (Wes McKinney 2010), while the normalizing flows are implemented in Jax (Bradbury et al. 2018), which allows for efficient, parallelizable, GPU-enabled calculations for very large data sets. The code is easily installable from the Python Package Index[8] (PyPI) and is hosted on Github,[9]. The documentation[10] includes tutorial notebooks demonstrating the features mentioned in this paper on different example problems.

The rest of this paper will demonstrate using PZFlow for the statistical modeling of galaxy catalogs. Section 4 uses PZFlow to forward model a galaxy catalog, including photometry, spectroscopic redshifts (spec-z's), *true* photo-z posteriors, ellipticities, and sizes. Section 5 uses PZFlow for photo-z estimation, demonstrating the power of PZFlow as a density estimator, including numerous useful features for photo-z estimation.

In addition to the examples in this paper, PZFlow has already being used in various other projects:

- Malz et al. (2021) used PZFlow to build a metric for observing strategy optimization based on information theory;
- Stylianou et al. (2022) used PZFlow to forward model galaxy data with true redshift posteriors in order to evaluate the impact of survey incompleteness and spec-z errors on photo-z estimation;
- Lokken et al. (2022) used PZFlow to smooth high-redshift artifacts in simulations of host galaxies for supernovae and other transients.

4. FORWARD MODELING A GALAXY CATALOG

In this section, we use PZFlow to forward model a photometric galaxy catalog for the Vera Rubin Observatory's Legacy Survey of Space and Time (LSST; Ivezić et al. 2019). The advantage of using a catalog generated from a normalizing flow is that we have direct access to the exact distribution from which the data is drawn, enabling us to calculate true values for derived statistical products, such as the *true* photo-z redshift posterior for each galaxy.

In Section 4.1 we construct a normalizing flow to model the galaxy redshifts and photometry and generate a new simulated catalog. In Section 4.2, we calculate true redshift posteriors for the new catalog. In Section 4.3 we build a conditional flow to add additional galaxy properties to the catalog.

4.1. *Forward modeling redshifts and photometry*

To create a generative model of galaxy redshifts and photometry, we use the true redshifts and *ugrizy* magnitudes from the CosmoDC2 simulation (LSST Dark Energy Science Collaboration et al. 2021; Korytov et al. 2019) of the LSST Dark Energy Science Collaboration (DESC). We selected all galaxies from CosmoDC2 with at most one band with a signal-to-noise ratio (SNR) less than 10, using the forecasted photometric errors from LSST year 10. Of these, we randomly selected 10^6 galaxies and split them intro training and test sets consisting of 80% and 20% of the galaxies, respectively.

For the latent distribution we use a 7 dimensional Uniform distribution over the range $[-5, 5]$[11]. To map the data onto the latent distribution, we use the following bijection:

$$f = \text{RQ-RSC} \circ \text{Shift Bounds} \circ \text{Color Transform}. \quad (15)$$

We will explain each layer of the bijection in the order they are applied to the input data.

The first layer of the bijection is the Color Transform, a data processing bijection designed specifically for this task. The Color Transform converts galaxy magnitudes to colors, but keeps the i band magnitude as a proxy for the apparent luminosity:

$$\text{Color Transform} : (\text{redshift}, u, g, r, i, z, y) \rightarrow (\text{redshift}, i, u-g, g-r, r-i, i-z, z-y). \quad (16)$$

This layer is useful as galaxy redshifts correlate more directly with galaxy colors than galaxy magnitudes.

The next layer, Shift Bounds, is the data processing bijection defined in Section 2.1.2, which maps the range of the data onto the support of the RQ-RSC. Note that since Shift Bounds is on the "other side" of the Color Transform, we need to map the ranges of the colors $u-g$, $g-r$, etc. onto the support of the splines, instead of the original magnitudes.

[8] https://pypi.org/project/pzflow/
[9] https://github.com/jfcrenshaw/pzflow
[10] https://jfcrenshaw.github.io/pzflow/

[11] The choice of 5 was arbitrary. Any other positive value would work just as well.

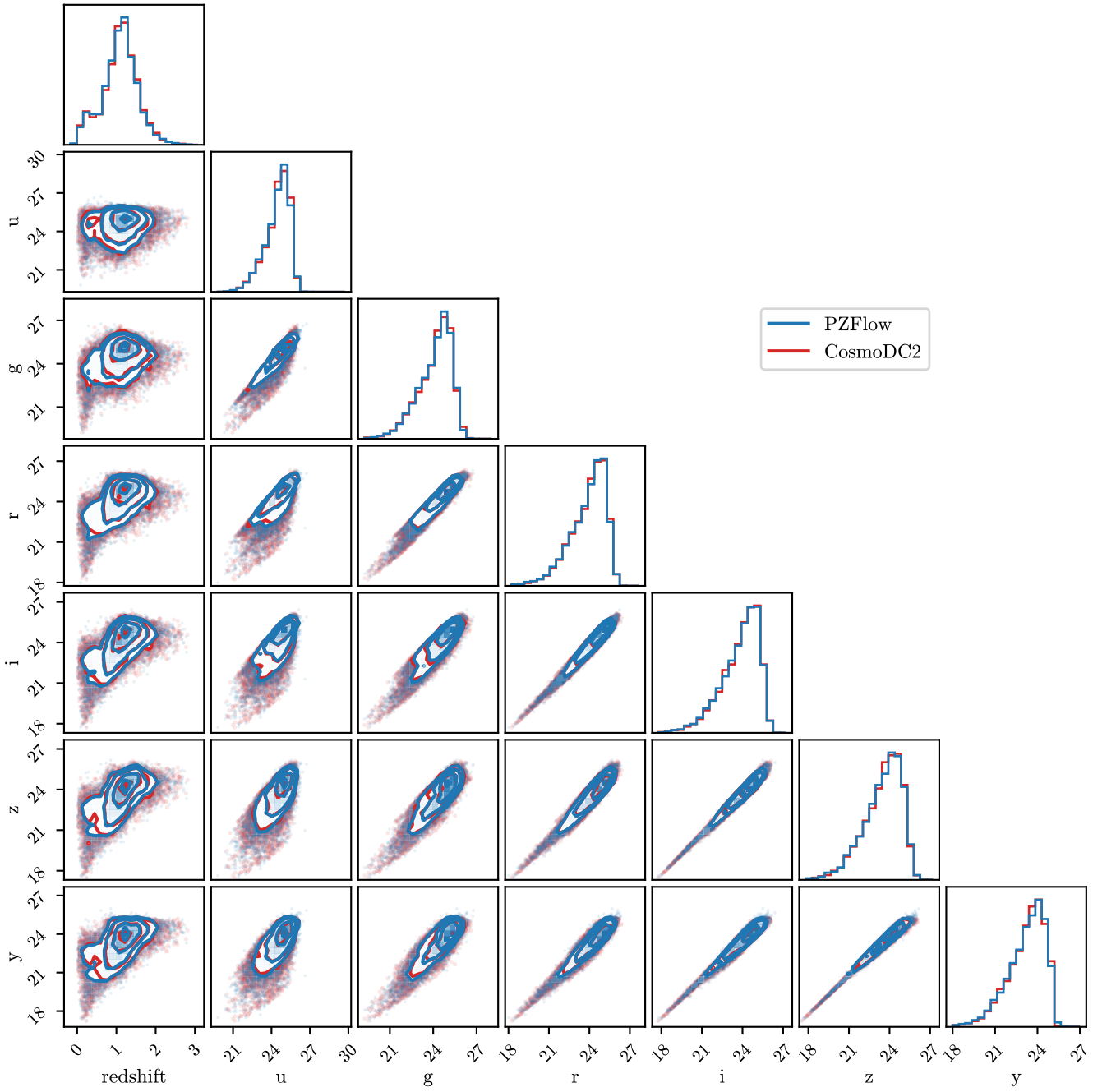

Figure 4. Distribution of true redshifts and noiseless photometry from the CosmoDC2 test set, compared to a sample drawn from the distribution learned by PZFlow. The close overlap of every pair-wise distribution demonstrates that PZFlow has learned the distribution in CosmoDC2 with high fidelity.

The final layer is an RQ-RSC, described in detail in Section 2.1.1. This layer performs the heavy lifting of transforming the data distribution into the uniform latent distribution. We use $D = 7$ layers to transform all 7 dimensions of our data, and set $B = 5$ to match the support of the latent distribution. We use the coupling function (a feedforward neural network with two hidden layers of 128 neurons) described in Durkan et al. (2019). We use $K = 16$ spline knots.

After training the flow (see Appendix A), we assess the results by drawing 10^4 galaxies from the trained flow, and plotting their distribution against 10^4 galaxies from the test set (Figure 4). We see the normalizing flow has done an excellent job of reproducing the distribution of galaxies in CosmoDC2, without any unusual artifacts or

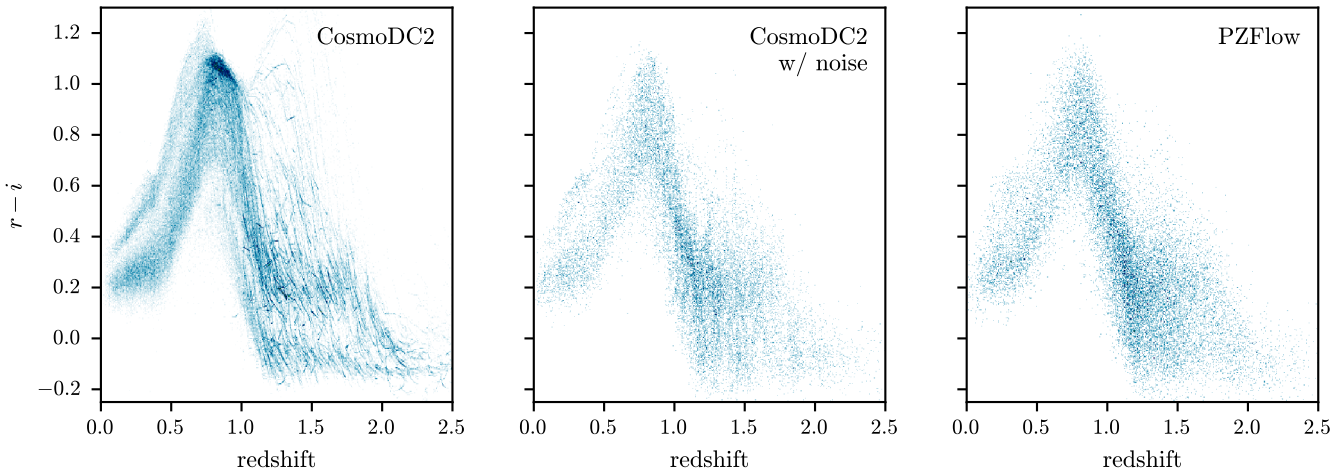

Figure 5. Comparing the noiseless $r-i$ vs redshift distribution for galaxy samples from CosmoDC2 (left) and from the normalizing flow (right). The high-redshift galaxies in CosmoDC2 lie along discrete tracks in color space due to the discrete number of galaxy SED templates used in the simulation. PZFlow smooths over these discrete tracks, resulting in a color distribution that is smooth to high redshifts.

outliers. In addition, Figure 5 compares the distribution of galaxy $r-i$ vs redshift. The CosmoDC2 simulation is known to exhibit discrete tracks in this space at high redshift, due to the discrete number of SED templates used during simulation. These tracks are visible in the left panel. The right panel shows that PZFlow smooths over this discreteness, resulting in a color distribution that is smooth up to high redshifts.

We note that these results were obtained without any extensive hyperparameter search, and that very similar (slightly worse results) are obtained without the `ColorTransform` bijection, demonstrating the flexibility of the method to adapt to unseen data sets.

With this normalizing flow, we have an efficient CosmoDC2 emulator that produces a smooth distribution of realistic galaxies up to high-redshifts. We use this emulator to generate a catalog with 10^4 galaxies. We add photometric errors using the LSST error model of our PhotErr package (see Appendix B). Importantly, since we have access to the probability distribution from which the galaxies were generated, we can calculate *true* redshift posteriors for each galaxy. This is the subject of the next section.

4.2. *Calculating true posteriors*

Since we have direct access to the probability distribution from which the photometry and redshifts are drawn, we can analytically calculate the true redshift posterior for each galaxy: $p(z|m)$ where m is the vector of galaxy magnitudes. We note that this is not an estimate, like what would be returned by a photo-z estimator, but rather the truth, obtained from the model that generated the photometry and redshifts in the first place.

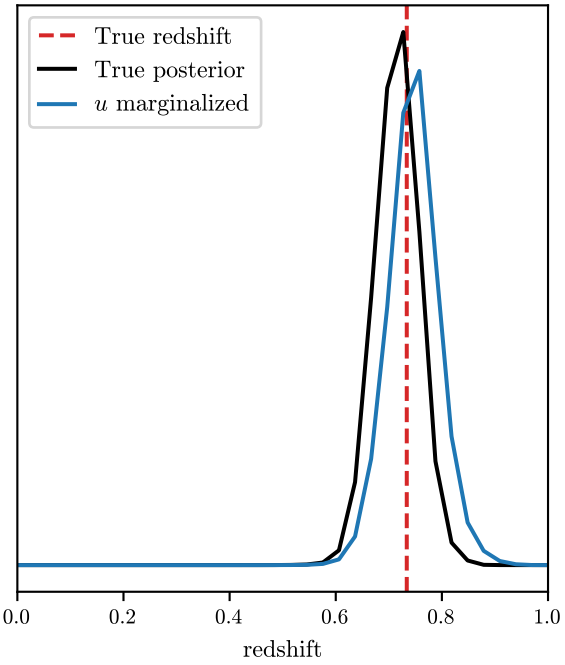

Figure 6. True redshift posteriors for a galaxy representing different amounts of information. The black posterior is calculated with the true magnitudes; the blue posterior is calculated after adding photometric errors; the orange posterior is calculated after adding photometric errors, but with the errors convolved during posterior calculation; the green posteriors is the same as the orange, except with the u band marginalized over.

In addition to calculating the true posterior, $p(z|m)$, we can calculate a true posterior that is consistent with

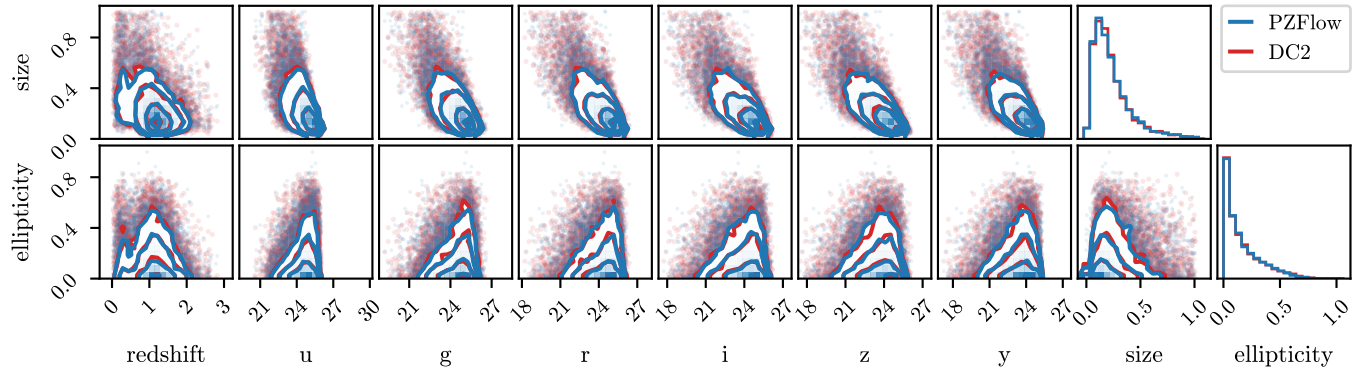

Figure 7. Conditional distributions of the ellipticity and size of the galaxies in the CosmoDC2 test set compared to the distribution learned by PZFlow. The close overlap of every pair-wise distribution demonstrates that PZFlow has learned the distribution in CosmoDC2 with high fidelity.

the photometric errors:

$$p(z|m,\sigma_m) = \int p(z|\hat{m})p(\hat{m}|m,\sigma_m)d\hat{m}, \qquad (17)$$

where σ_m is the vector of photometric errors returned by the error model, and $\hat{m}$ are possible noisy observations of the true magnitudes, m. We can evaluate this integral numerically by sampling $\hat{m} \sim p(\hat{m}|m,\sigma_m)$, evaluating $p(z,\hat{m})$ on a redshift grid, and normalizing to obtain $p(z|\hat{m})$. Averaging $p(z|\hat{m})$ over the samples $\hat{m}$ yields $p(z|m,\sigma_m)$. This is the "best case scenario" redshift posterior that photo-z estimators hope to estimate.

We can also marginalize over any missing bands, n:

$$p(z|\hat{m}) = \frac{1}{p(\hat{m})}\int p(z,n,\hat{m})dn, \qquad (18)$$

which can be calculated by evaluating $p(z,n,\hat{m})$ on a grid of z and n, summing over n to yield $p(z,\hat{m})$, and normalizing with respect to redshift to yield $p(z|\hat{m})$. You can once again average over $\hat{m}$ samples to convolve the photometric errors. You may wish to marginalize over all values of n if the galaxy was not observed in that band. This might occur, for example, if you include photometry from the Euclid Space Telescope (Scaramella et al. 2022), which will not have complete coverage of the LSST catalog. You may also wish to marginalize over all values beyond a limiting magnitude if the galaxy was observed, but not detected in that band. This might occur, for example, in the low wavelength bands of Lyman dropout galaxies observed by LSST.

To visualize the impacts of error convolution and band marginalization, Figure 6 shows a number of redshift posteriors for a single example galaxy. The black posterior is calculated using the true galaxy magnitudes, while the blue posteriors is calculated after adding photometric errors. Calculating the posterior using the noisy photometry results in a biased posterior. The orange posterior has had the photometric errors convolved as in Equation 17. Convolving the errors broadens the posterior so that the true redshift lies within the support of the posterior. This broadening reflects the increased uncertainty due to the photometric errors. Finally, the green posterior has had the u band marginalized. This posterior favors higher redshifts, but still comfortably agrees with the true redshift. This demonstrates how the u band helps constrain the redshift, and the loss of this information leads to greater uncertainty.

Calculating these posteriors enables direct comparison of true redshift posteriors (consistent with photometric errors and missing bands) with the redshift posteriors estimated by photo-z estimators. This is important, as modern cosmology analyses are beginning to increasingly rely on full redshift posteriors (Mandelbaum et al. 2008; Newman & Gruen 2022). Schmidt & Malz et al. (2020) showed that popular metrics for evaluating photo-z estimators using ensembles of photo-z posteriors can be misleading, and are not well suited to our science. PZFlow catalogs with *true* redshift posteriors provide a path forward by enabling the evaluation of photo-z estimators on a per-posterior basis.

4.3. *Additional properties with conditional flows*

In addition to the galaxy magnitudes and redshifts modeled above, we wish to include other galaxy properties in the catalog, such as galaxy size and ellipticity. In principle, we could have included these variables in the original normalizing flow. However, we did not want the true redshift posteriors to be conditioned on these variables, as most photo-z estimators only use galaxy photometry. Therefore, we will build a second flow that models these additional values conditioned on the galaxy redshift and magnitudes. Note that while we have only

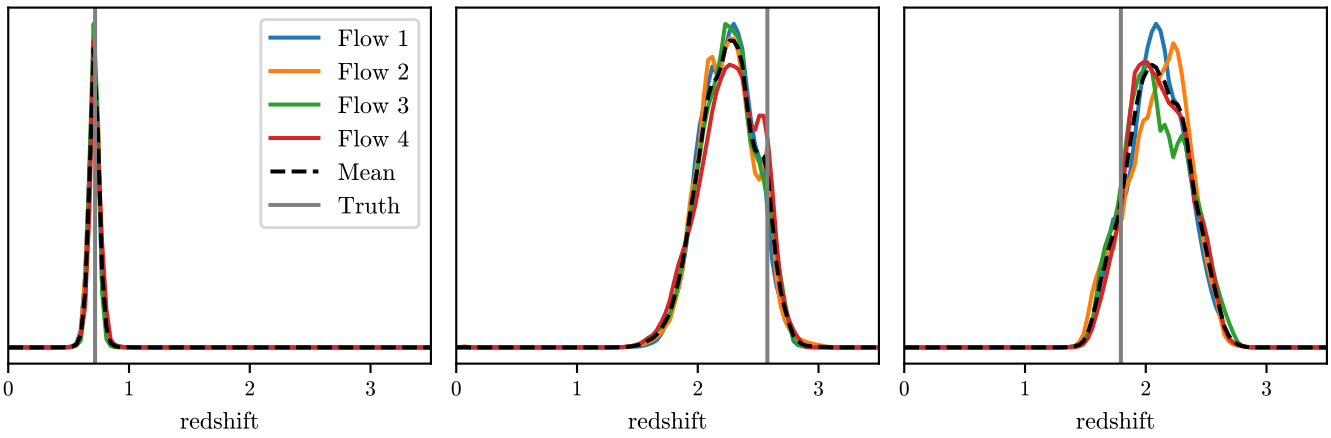

Figure 8. The ensemble of posteriors for three example galaxies. Flows 1-4 label the individual posteriors produced by each of the flows that make up the ensemble. The dashed black line is the mean of these individual posteriors and is the value used by the ensemble. The vertical gray line labeled "Truth" denotes the true redshift of the galaxy. Note these galaxies were specifically chosen for their broad, multimodal posteriors. The posteriors of most galaxies are sharp and unimodal.

chosen to model these additional two properties, any other values you desire can be similarly modeled.

For the latent distribution, we again use a Uniform distribution over the range $[-5, 5]$. For the bijection, we use

$$f = \text{RQ-RSC} \circ \text{Shift Bounds}. \tag{19}$$

The RQ-RSC acts on the two dimensional space of size and ellipticity, but also takes the galaxy redshift and magnitudes as inputs (see Figure 3). The redshifts and magnitudes are transformed to have zero mean and unit variance before being input to the neural network[12] that parameterizes the splines. Aside from the change in inputs, the RQ-RSC has the same settings as listed for the previous normalizing flow.

After training the flow (see Appendix A), we sample a size and ellipticity for each galaxy in the PZFlow catalog created in the previous section (conditioned on the true magnitudes), and plot the distribution of these features against the distribution in the test set (Figure 7). Once again, we see the normalizing flow does a good job of emulating the CosmoDC2 galaxy distribution.

The final simulated catalog consists of 10^4 galaxies, each with a redshift, *ugrizy* magnitudes including photometric errors, a true photo-z posterior consistent with the photometric errors, a size, and an ellipticity. This small catalog was generated for visualization purposes, but the normalizing flows can be used to generate catalogs of arbitrarily large size.

[12] These variables are standard scaled instead of mapped onto the domain [-5, 5], because the neural network that parameterizes the splines has no limit on inputs, unlike the splines themselves, which are limited to the range [-5, 5].

5. PHOTOMETRIC REDSHIFT ESTIMATION

In addition to forward modeling, normalizing flows are powerful and flexible models for density estimation. This makes them useful tools for estimating posterior distributions for galaxy properties, conditioned on observed features of the galaxy. A common example of this in cosmology is photometric redshift estimation, in which you estimate the redshift of a galaxy using its magnitude in several photometric bands. In this section, we use PZFlow as a photo-z estimator to demonstrate using normalizing flows for density estimation.

5.1. *Training an Ensemble for photo-z estimation*

When forward modeling in Section 4, we wanted a realistic model that captured the relevant correlations between galaxy photometry, redshift, shape, and size. However, when estimating redshifts, we do not simply want a realistic model, but rather a model that matches our specific galaxy sample as closely as possible.

When training deep learning models, the huge parameter space contains many different solutions, corresponding to different local minima in the parameter space. In the forward modeling application, we were content with finding a good local minimum, but in this application, we want to marginalize over the different potential models.

A full marginalization over the model parameters would be too computationally expensive, so instead we approximate this marginalization using an ensemble of normalizing flows. In other words, we train multiple normalizing flows under identical conditions, using different random initializations of the model parameters. This allows the optimization algorithm to explore different basins of attraction in the parameter space. In

the machine learning literature, this is known as a Deep Ensemble (Lakshminarayanan et al. 2017), and is a popular method for approximate bayesian marginalization (Wilson & Izmailov 2020; Fort et al. 2020).

We train an ensemble of 4 normalizing flows, each with the same architecture and training schedule as the regular flow described in Section 4. With PZFlow, this is as simple as swapping `FlowEnsemble` for `Flow` in the code.

For the training set, we use 100,000 galaxies from the catalog created in Section 4. Each galaxy in the training set has a true redshift and observed magnitudes in the *ugrizy* bands, with corresponding photometric errors. To account for the photometric error, at the start of each training epoch, we resample the training set from the photometric error distributions. In other words, each epoch, for each galaxy, we sample

$$m \sim p(\hat{m}, \sigma_m), \tag{20}$$

where $\hat{m}$ are the observed magnitudes with photometric errors σ_m, and $p(\hat{m}, \sigma_m)$ is a Gaussian in flux space. This allows our ensemble of flows to approximate the distribution $p(z, m)$. For more details on training the ensemble, see Appendix A.

5.2. *Estimating posteriors*

For each flow in the ensemble, we estimate $p(z, m)$ by marginalizing over the photometric errors:

$$p(z|\hat{m}, \sigma_m) \propto \int dm\, p(z, m)\, p(m|\hat{m}, \sigma_m), \tag{21}$$

which is estimated by sampling $m \sim p(\hat{m}, \sigma_m)$ and averaging $p(z, m)$ over these samples. We then average the $p(z, m)$ from each flow, and normalize with respect to the redshift grid. This provides a redshift posterior for each galaxy.

Posteriors for three galaxies can be seen in Figure 8. Each flow produces a PDF which may contain slightly different features in each case. By averaging over the individual posteriors, we select for features that are common between models, while smoothing over features that are present in only a single model. We can also treat the ensemble of posteriors as a distribution over possible posteriors, which will allow for more consistent error calibration in cosmological analyses (Zhang et al. 2023).

We can calculate posteriors for galaxies with missing magnitudes by marginalizing over the missing magnitudes, as described in Section 4.2. An example is shown in Figure **??**, which compares a posterior calculated using every LSST band to a posterior with the u band marginalized. Sacrificing the information in the u band increases the uncertainty of the redshift inference. In this instance, a second potential higher-redshift mode is created. Conversely, this demonstrates how the addition of u band information rules out the higher redshift solution for this galaxy.

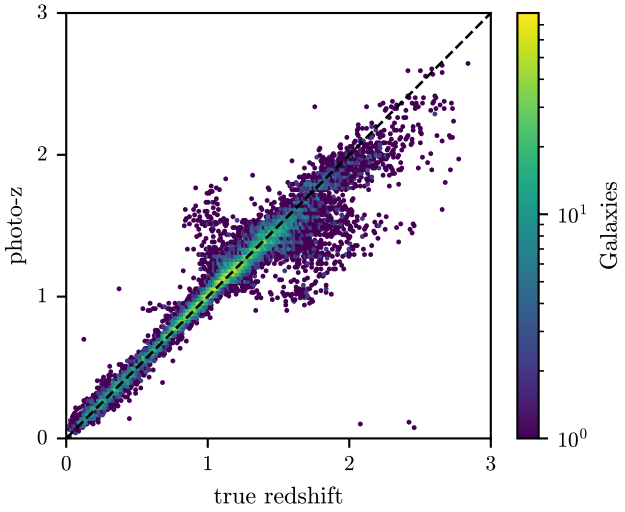

Figure 9. Photo-z point estimates (maximum a posteriori) vs true redshift for galaxies in the test set.

5.3. *Photo-z metrics*

In this section, we evaluate the performance of PZFlow using common photo-z metrics. Note these metrics are optimistic in the sense that the training set is representative of the test set, which is usually not the case in modern cosmology applications.

The most common metrics for photo-z estimation concern photo-z point estimates, which are a compression of the photo-z posterior to a single redshift estimate (e.g., Hildebrandt et al. 2010; Sánchez et al. 2014). We make the common choice of selecting the mode of the posteriors[13]. We compute metrics of the quantity $\Delta z = (z_{\rm phot} - z_{\rm true})/(1 + z_{\rm true})$, where the denominator accounts for naturally greater uncertainties at high redshift.

Figure 9 compares the photo-z point estimates to the true redshifts. The point estimates for most galaxies lie along the diagonal, indicating strong performance. There are the common photo-z "wings", indicating redshifts where important spectral features are transitioning between neighboring photometric bands. There is also a small population of high-redshift objects that were mistakenly identified as very low redshift objects. This point estimate plot is comparable to other high-performance machine learning photo-z esti-

[13] The mean redshift is a poor choice, since photo-z posteriors are often multimodal, and so the mean value can lie between two modes at a redshift with very small probability density.

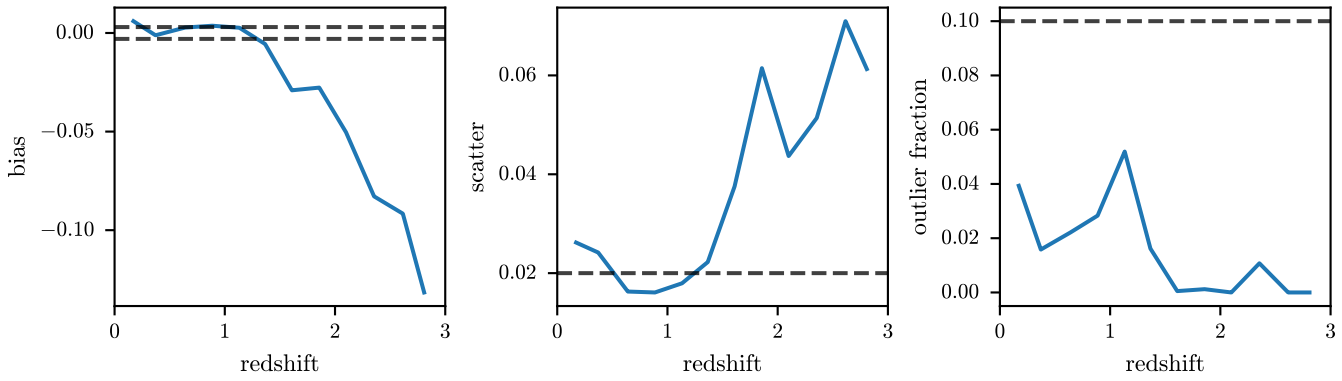

Figure 10. The bias, scatter, and outlier fraction of the photo-z point estimates as a function of true galaxy redshift. The dashed black lines represent the requirements for LSST cosmology as stated in the LSST DESC SRD (The LSST Dark Energy Science Collaboration et al. 2018). These lines are to provide a sense of scale for these metrics. You can see that PZFlow meets the bias and scatter requirements up to redshift ~ 1.5, while meeting the outlier fraction requirements for all redshifts. We note that individual redshifts do not actually need to meet the bias requirement as long as the bias can be well calibrated via some other source, e.g. galaxy clustering.

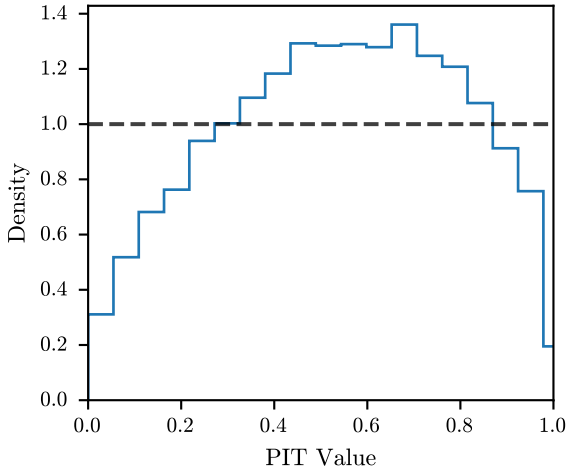

Figure 11. The probability integral transform (PIT) histogram for PZFlow photo-z posteriors. The PIT characterizes the calibration of the estimated posteriors, with the horizontal black line indicating perfect calibration.

mators when provided with representative training sets (Sánchez et al. 2014).

Figure 10 shows the photo-z point estimate metrics from the LSST DESC Science Requirements Document (SRD; The LSST Dark Energy Science Collaboration et al. 2018) as a function of true redshift. The *bias* is defined as the median of Δz; the *scatter* is defined as IQR/1.349, where IQR is the interquartile range of Δz; the *outlier fraction* is defined as the fraction of galaxies for which Δz is greater than three times the scatter. The requirements from the SRD are plotted in black to provide a sense of scale.

Like many photo-z estimators, PZFlow performs well to a redshift of approximately 1.5. At higher redshifts, our estimator does not meet the bias and scatter requirements, because there is very little training data in this redshift range. We note however that for many cosmology applications, it is okay for the bias to exceed the required limits, as long as the bias can be well determined via some calibration process (Newman et al. 2015).

Another common metric is the probability integral transform (PIT) (see e.g. Schmidt & Malz et al. 2020, Dey et al. 2022). The PIT metric is used to determine if posteriors are well calibrated, i.e. if true values fall within X% confidence intervals X% of the time. The PIT histogram for our estimator is shown in Figure 11. Ideally, this histogram would be uniform and match the dashed horizontal line. The fact that the histogram bulges at the center indicates that our estimator is too conservative – i.e. the posteriors it produces are too broad. This can be explained by the fact that normalizing flows exhibit mode covering behavior (the opposite of the mode collapse seen in GANs; Salimans et al. 2016). In other words, because normalizing flows are trained by maximizing the likelihood of the training samples, they receive very high penalties for missing any modes in the data. As a result, they tend to conservatively spread out their density, in order to avoid missing any modes. This results in overly conservative posterior predictions. The low values at the edges of the PIT histogram indicate the relative rarity of catastrophic outliers, which is also reflected in the far right panel of Figure 10, where you can see that our estimator meets the requirement on the outlier fraction at all redshifts. There is also a slight rightward tilt. This indicates a small negative bias, which reflects the intrinsic prior towards smaller redshifts, as this is where the ma-

jority of galaxies in the training set lie. This negative bias is also visible in the far left panel of Figure 10.

The previous metrics analyze photo-z performance for point estimates, which are insufficient for modern cosmology (Newman & Gruen 2022), and for ensembles of posteriors, which is often misleading and not a good indicator of performance for science applications (Schmidt & Malz et al. 2020). The methods introduced in this paper enable the creation of galaxy catalogs for which each galaxy has a *true* redshift posterior, which will enable more comprehensive evaluation of photo-z estimators. An full evaluation of photo-z estimators on a posterior-by-posterior basis is a major goal of the LSST DESC.

6. CONCLUSION & SUMMARY

In this paper we introduced PZFlow, a normalizing flow package in Python, designed for the statistical modeling of tabular astronomical data. We used PZFlow to forward model galaxy catalogs that include photometry, redshifts, sizes, and ellipticities. In addition, each galaxy in our catalog has a *true* redshift posterior, which can be convolved with measurement errors. These true posteriors allow a direct evaluation of the posteriors produced by photo-z estimators. A similar comparison can be made to posterior estimates for any other galaxy properties.

Direct evaluation of photo-z posteriors is vital for future cosmology analyses which must use all of the information incorporated in the full redshift posteriors (Newman & Gruen 2022). This is because only the full redshift posteriors allow one to account for degeneracies in color-redshift space, which will otherwise bias cosmological inference. Previous evaluations of photo-z performance have focused on point estimates and metrics for ensembles of posteriors, but Schmidt & Malz et al. (2020) demonstrated these metrics are misleading and inadequate for modern cosmology. This work enables future analysis of photo-z estimators on a per-posterior basis, which is a major goal of the LSST DESC.

In addition to forward modeling galaxy catalogs, we demonstrated PZFlow's utility as a density estimator that can be applied to photo-z estimation and other statistical analyses. PZFlow achieves high accuracy with relatively little fine tuning and with very few modeling assumptions. This makes PZFlow a powerful tool for the statistical analysis of tabular astronomical data.

This paper was written using the showyourwork[14] workflow manager. The code to reproduce this paper is hosted publicly on Github[15], and the code for each individual figure can be found by clicking on the Github logo in the margin next to that figure.

Thanks to François Lanusse for some early advice on normalizing flows and for reviewing the paper, and to Martine Lokken for testing PZFlow. J. F. Crenshaw, B. J. Kalmbach, and A. J. Connolly acknowledge support from the DiRAC Institute in the Department of Astronomy at the University of Washington. The DIRAC Institute is supported through generous gifts from the Charles and Lisa Simonyi Fund for Arts and Sciences, and the Washington Research Foundation. Z. Yan acknowledges support from the Max Planck Society and the Alexander von Humboldt Foundation in the framework of the Max Planck-Humboldt Research Award endowed by the German Federal Ministry of Education and Research. AIM acknowledges support during the course of this work from the Max Planck Society and the Alexander von Humboldt Foundation in the framework of the Max Planck-Humboldt Research Award endowed by the Federal Ministry of Education and Research.

Author contributions are listed below:
J. F. Crenshaw: created PZFlow, designed experiments, wrote paper.
J. B. Kalmbach: wrote early normalizing flow code that evolved into PZFlow; photo-z estimation.
A. Gagliano: validated code, developed use cases and associated tutorials; revised manuscript text.
Z. Yan: contributed to PZFlow and PhotErr.
A. J. Connolly: discussion during development.
A. I. Malz: consulted on design and testing of PZFlow.
S. J. Schimdt: consulted on design and testing of PZFlow and PhotErr.

Software: adam (Kingma & Ba 2015), corner (Foreman-Mackey 2016), dill (McKerns et al. 2012), jax (Bradbury et al. 2018), jupyter (Kluyver et al. 2016), matplotlib (Hunter 2007), numpy (Harris et al. 2020), pandas (Wes McKinney 2010; Reback et al. 2020), scipy (Virtanen et al. 2020), showyourwork (Luger et al. 2021), scikit-learn (Pedregosa et al. 2011)

APPENDIX

[14] https://show-your.work/

[15] https://github.com/jfcrenshaw/pzflow-paper

A. TRAINING DETAILS

In this section we list some technical details of training the normalizing flows. Every flow is trained via minimizing the negative log-likelihood

$$\mathcal{L} = -\,\mathbb{E}[\,\log p(x)\,], \tag{A1}$$

where the expectation is performed over galaxies in the training set and $p(x)$ is defined in Equation 1.

For the main flow in Section 4, we trained for 150 epochs. We used the Adam optimizer (Kingma & Ba 2015), starting with a learning rate of 10^{-3}. We decreased the learning rate by a factor of 10 every 50 epochs. Training took 7 minutes on a Tesla P100 12GB GPU. The training loss for this flow is in the left panel of Figure 12.

For the conditional flow in 4, we trained for 450 epochs. Again, we used Adam with an initial learning rate of 10^{-3}. We decreased the learning rate by a factor of 10 every 150 epochs. The training loss for this flow is in the right panel of Figure 12.

For each of the flows that make up the flow ensemble in Section 5, we trained for 150 epochs using the Adam optimizer. We started each with a learning rate of 10^{-4}, and decreased the learning rate by a factor of 10 every 50 epochs. The training loss for the ensemble is in Figure 13. You can see that each of the flows achieves a different minimum loss. Apparently, each has found a different potential solution in the neural network's parameter space.

B. LSST ERROR MODEL

We estimate photometric errors for LSST using a generalization of the error model from Ivezić et al. (2019). To derive the error model, we start with the noise-to-signal ratio (NSR) for an object with photon count C and background noise N_0 (which depends on seeing, read-out noise, etc.):

$$\mathrm{NSR}^2 = \frac{N_0^2 + C}{C^2}. \tag{B2}$$

If we define $C = C_5$ when NSR $= 1/5$, then we can solve for N_0 and write

$$\mathrm{NSR}^2 = \frac{1}{C_5}\left(\frac{C_5}{C}\right) + \left[\left(\frac{1}{5}\right)^2 - \frac{1}{C_5}\right]\left(\frac{C_5}{C}\right)^2. \tag{B3}$$

Defining $x = C_5/C = 10^{(m-m_5)/2.5}$ and $\gamma = 1/5^2 - 1/C_5$, we have

$$\mathrm{NSR}^2 = (0.04 - \gamma)\,x + \gamma\,x^2 \quad (\mathrm{mag}^2), \tag{B4}$$

which is Equation 5 from Ivezić et al. (2019). Values for the band-dependent parameter γ can be found in Table 2 of the same paper.

In the high signal-to-noise (SNR) limit, NSR $\ll 1$, and we can approximate

$$\sigma_{\mathrm{rand}} = 2.5\log_{10}(1 + \mathrm{NSR}) \approx \mathrm{NSR}. \tag{B5}$$

This latter approximation is made by Ivezić et al. (2019), and errors are assumed to be Gaussian in magnitude space. In contrast, we use the exact form of Equation B5, and model errors as Gaussian in flux space. Note that after the photometric errors are applied, the error is re-calculated from the "observed" flux, and this new error is reported as the estimated photometric error. If the original photometric error were reported, it would provide a deterministic link to the original flux.

We have implemented this error model, along with several other extensions, in the Python package PhotErr, which is available on the Python Package Index[16] (PyPI), and Github[17]. The extensions include different methods for handling non-detections, methods for modeling errors of extended objects (using models from van den Busch et al. 2020; Kuijken et al. 2019), and error models for the Roman and Euclid space telescopes (Spergel et al. 2015; Scaramella et al. 2022; Graham et al. 2020).

[16] https://pypi.org/project/photerr/

[17] https://github.com/jfcrenshaw/photerr

REFERENCES

Bradbury, J., Frostig, R., Hawkins, P., et al. 2018, JAX: Composable Transformations of Python+NumPy Programs. http://github.com/google/jax

Dacunha, T., Raveri, M., Park, M., Doux, C., & Jain, B. 2022, PhRvD, 105, 063529, doi: 10.1103/PhysRevD.105.063529

Dai, B., & Seljak, U. 2022, MNRAS, 516, 2363, doi: 10.1093/mnras/stac2010

Dey, B., Zhao, D., Newman, J. A., et al. 2022, Calibrated Predictive Distributions via Diagnostics for Conditional Coverage, doi: 10.48550/arXiv.2205.14568

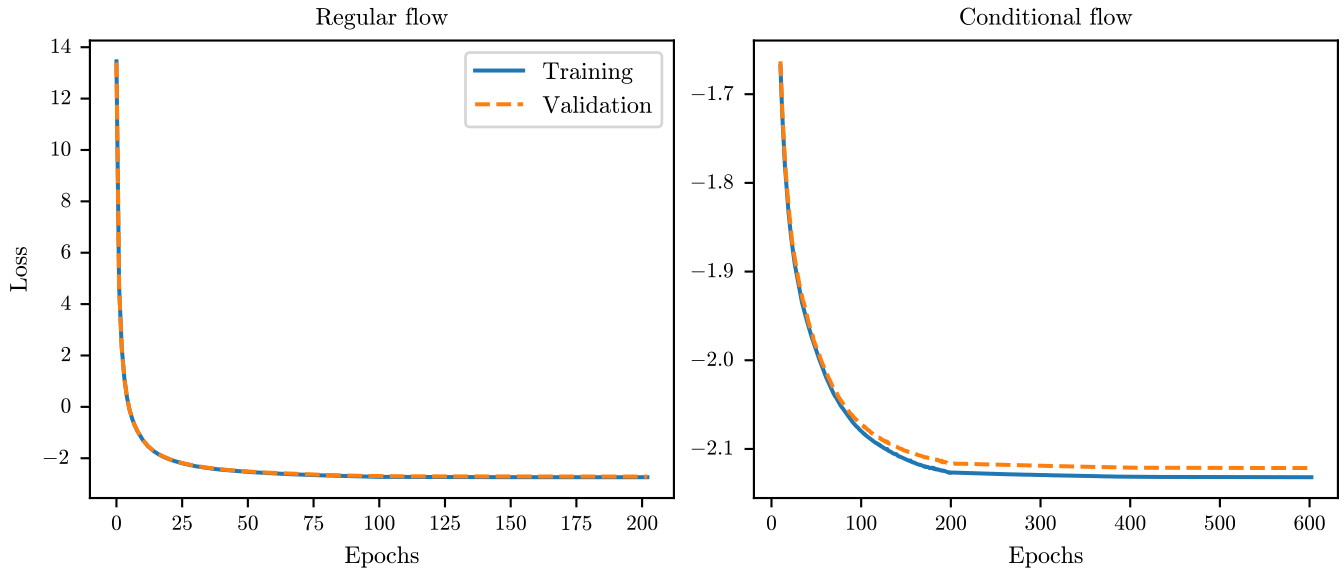

Figure 12. Training losses for the normalizing flows. Left: losses for the regular flow. After epochs 50 and 100, you can see a drop in the loss due to the decrease in the learning rate. Right: losses for the conditional flow. After epochs 150 and 300, you can see a drop in the loss due to the decrease in the learning rate.

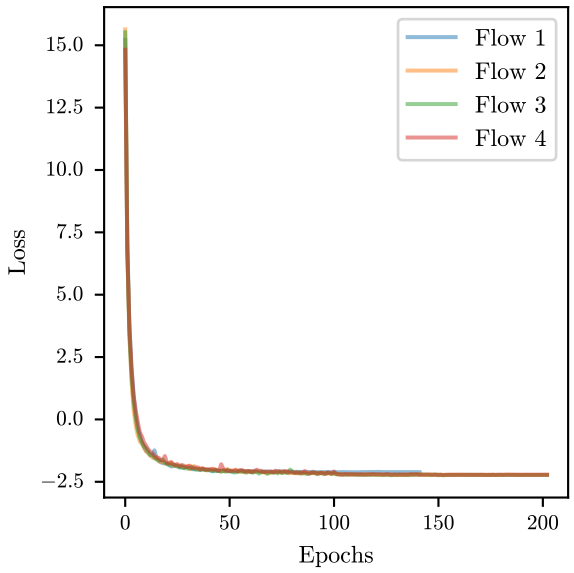

Figure 13. Training losses for the four flows in the flow ensemble. We have zoomed in to the bottom of the loss curve so you can see that each of the flows converges to a different minimum loss.

Dinh, L., Krueger, D., & Bengio, Y. 2015, in Proceedings of the 3rd International Conference on Learning Representations, ed. Y. Bengio & Y. LeCun, San Diego, CA. https://arxiv.org/abs/1410.8516

Dinh, L., Sohl-Dickstein, J., & Bengio, S. 2017, in Proceedings of the 5th International Conference on Learning Representations, Toulon, France. https://arxiv.org/abs/1605.08803

Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. 2019, in Advances in Neural Information Processing Systems 32, ed. H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Vancouver, Canada: Curran Associates, Inc.), 7511–7522. https://arxiv.org/abs/1906.04032

Falorsi, L., de Haan, P., Davidson, T. R., & Forré, P. 2019, in Proceedings of Machine Learning Research, Vol. 89, The 22nd International Conference on Artificial Intelligence and Statistics, ed. K. Chaudhuri & M. Sugiyama (Naha, Okinawa, Japan: PMLR), 3244–3253. http://proceedings.mlr.press/v89/falorsi19a.html

Foreman-Mackey, D. 2016, The Journal of Open Source Software, 1, 24, doi: 10.21105/joss.00024

Fort, S., Hu, H., & Lakshminarayanan, B. 2020, arXiv:1912.02757 [cs, stat]. https://arxiv.org/abs/1912.02757

Gemici, M. C., Rezende, D. J., & Mohamed, S. 2016, CoRR, abs/1611.02304. https://arxiv.org/abs/1611.02304

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., et al. 2014, CoRR, abs/1406.2661, doi: 10.48550/arXiv.1406.2661

Graham, M. L., Connolly, A. J., Ivezić, Ž., et al. 2018, The Astronomical Journal, 155, 1, doi: 10.3847/1538-3881/aa99d4

Graham, M. L., Connolly, A. J., Wang, W., et al. 2020, The Astronomical Journal, 159, 258, doi: 10.3847/1538-3881/ab8a43

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Nature, 585, 357, doi: 10.1038/s41586-020-2649-2

Hassan, S., Villaescusa-Navarro, F., Wandelt, B., et al. 2022, ApJ, 937, 83, doi: 10.3847/1538-4357/ac8b09

Hildebrandt, H., Arnouts, S., Capak, P., et al. 2010, A&A, 523, A31, doi: 10.1051/0004-6361/201014885

Ho, J., Chen, X., Srinivas, A., Duan, Y., & Abbeel, P. 2019, in Proceedings of Machine Learning Research, Vol. 97, Proceedings of the 36th International Conference on Machine Learning, ed. K. Chaudhuri & R. Salakhutdinov (Long Beach, CA, USA: PMLR), 2722–2730. http://proceedings.mlr.press/v97/ho19a.html

Hoogeboom, E., Cohen, T. S., & Tomczak, J. M. 2020, CoRR, abs/2001.11235. https://arxiv.org/abs/2001.11235

Hunter, J. D. 2007, Computing in Science & Engineering, 9, 90, doi: 10.1109/MCSE.2007.55

Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, The Astrophysical Journal, 873, 111, doi: 10.3847/1538-4357/ab042c

Jaini, P., Kobyzev, I., Yu, Y., & Brubaker, M. 2020, in Proceedings of Machine Learning Research, Vol. 119, arXiv:1907.04481 [Cs, Math, Stat] (Virtual: PMLR), 4673–4681. https://arxiv.org/abs/1907.04481

Kessler, R., Narayan, G., Avelino, A., et al. 2019, Publications of the Astronomical Society of the Pacific, 131, 094501, doi: 10.1088/1538-3873/ab26f1

Kingma, D. P., & Ba, J. 2015, in 3rd International Conference on Learning Representations, ed. Y. Bengio & Y. LeCun, San Diego, CA. http://arxiv.org/abs/1412.6980

Kingma, D. P., & Welling, M. 2014, in 2nd International Conference on Learning Representations, ed. Y. Bengio & Y. LeCun, Banff, AB, Canada, doi: 10.48550/arXiv.1312.6114

Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, in Positioning and Power in Academic Publishing: Players, Agents and Agendas, ed. F. Loizides & B. Scmidt (Netherlands: IOS Press), 87–90. https://eprints.soton.ac.uk/403913/

Kobyzev, I., Prince, S. J. D., & Brubaker, M. A. 2020, IEEE Trans. Pattern Anal. Mach. Intell., 1, doi: 10.1109/TPAMI.2020.2992934

Korytov, D., Hearin, A., Kovacs, E., et al. 2019, The Astrophysical Journal Supplement Series, 245, 26, doi: 10.3847/1538-4365/ab510c

Kuijken, K., Heymans, C., Dvornik, A., et al. 2019, A&A, 625, A2, doi: 10.1051/0004-6361/201834918

Lakshminarayanan, B., Pritzel, A., & Blundell, C. 2017, in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, ed. I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett, 6402–6413. https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html

Lokken, M., Gagliano, A., Narayan, G., et al. 2022, The Simulated Catalogue of Optical Transients and Correlated Hosts (SCOTCH). https://ui.adsabs.harvard.edu/abs/2022arXiv220602815L

LSST Dark Energy Science Collaboration, Abolfathi, B., Alonso, D., et al. 2021, The Astrophysical Journal Supplement Series, 253, 31, doi: 10.3847/1538-4365/abd62c

Luger, R., Bedell, M., Foreman-Mackey, D., et al. 2021, Mapping Stellar Surfaces III: An Efficient, Scalable, and Open-Source Doppler Imaging Model. https://ui.adsabs.harvard.edu/abs/2021arXiv211006271L

Malz, A. I., Lanusse, F., Crenshaw, J. F., & Graham, M. L. 2021, An Information-Based Metric for Observing Strategy Optimization, Demonstrated in the Context of Photometric Redshifts with Applications to Cosmology, doi: 10.48550/arXiv.2104.08229

Mandelbaum, R., Seljak, U., Hirata, C. M., et al. 2008, Monthly Notices of the Royal Astronomical Society, 386, 781, doi: 10.1111/j.1365-2966.2008.12947.x

McKerns, M. M., Strand, L., Sullivan, T., Fang, A., & Aivazis, M. A. G. 2012, CoRR, abs/1202.1056. http://arxiv.org/abs/1202.1056

Newman, J. A., & Gruen, D. 2022, Annu. Rev. Astron. Astrophys., 60, annurev, doi: 10.1146/annurev-astro-032122-014611

Newman, J. A., Abate, A., Abdalla, F. B., et al. 2015, Astroparticle Physics, 63, 81, doi: 10.1016/j.astropartphys.2014.06.007

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12, 2825. http://jmlr.org/papers/v12/pedregosa11a.html

Reback, J., McKinney, W., jbrockmendel, et al. 2020, Pandas-Dev/Pandas: Pandas 1.0.3, Zenodo. https://doi.org/10.5281/zenodo.3715232

Rezende, D. J., Papamakarios, G., Racanière, S., et al. 2020, in Proceedings of Machine Learning Research, Vol. 119, arXiv:2002.02428 [Cs, Stat] (Virtual: PMLR), 8083–8092. https://arxiv.org/abs/2002.02428

Salimans, T., Goodfellow, I. J., Zaremba, W., et al. 2016, in Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, ed. D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, & R. Garnett, 2226–2234. https://proceedings.neurips.cc/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html

Sánchez, C., Carrasco Kind, M., Lin, H., et al. 2014, Monthly Notices of the Royal Astronomical Society, 445, 1482, doi: 10.1093/mnras/stu1836

Scaramella, R., Amiaux, J., Mellier, Y., et al. 2022, A&A, 662, A112, doi: 10.1051/0004-6361/202141938

Schmidt, S. J., Malz, A. I., Soo, J. Y. H., et al. 2020, arXiv:2001.03621 [astro-ph]. https://arxiv.org/abs/2001.03621

Spergel, D., Gehrels, N., Baltay, C., et al. 2015, Wide-Field InfrarRed Survey Telescope-Astrophysics Focused Telescope Assets WFIRST-AFTA 2015 Report, doi: 10.48550/arXiv.1503.03757

Stylianou, N., Malz, A. I., Hatfield, P., Crenshaw, J. F., & Gschwend, J. 2022, Publications of the Astronomical Society of the Pacific, 134, 044501, doi: 10.1088/1538-3873/ac59bf

The LSST Dark Energy Science Collaboration, Mandelbaum, R., Eifler, T., et al. 2018, arXiv e-prints, 1809, arXiv:1809.01669. http://adsabs.harvard.edu/abs/2018arXiv180901669T

van den Busch, J. L., Hildebrandt, H., Wright, A. H., et al. 2020, Astronomy and Astrophysics, 642, A200, doi: 10.1051/0004-6361/202038835

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: 10.1038/s41592-019-0686-2

Wes McKinney. 2010, in Proceedings of the 9th Python in Science Conference, ed. S. van der Walt & Jarrod Millman, 56–61, doi: 10.25080/Majora-92bf1922-00a

Wilson, A. G., & Izmailov, P. 2020, arXiv:2002.08791 [cs, stat]. https://arxiv.org/abs/2002.08791

Winkler, C., Worrall, D., Hoogeboom, E., & Welling, M. 2019, CoRR, abs/1912.00042. https://arxiv.org/abs/1912.00042

Zhang, T., Rau, M. M., Mandelbaum, R., Li, X., & Moews, B. 2023, Monthly Notices of the Royal Astronomical Society, 518, 709, doi: 10.1093/mnras/stac3090