# Undergraduate Thesis: Inductive Bias and Uncertainty Calibration in the era of JWST NIRCam Science

**Edward Berman** *berman.ed@northeastern.edu*
*Northeastern University*

## Abstract

This document contains the collection of my undergraduate papers in the form of an (unofficial) undergradute thesis. This "thesis" is dedicated to my high school physics teacher, Doc.

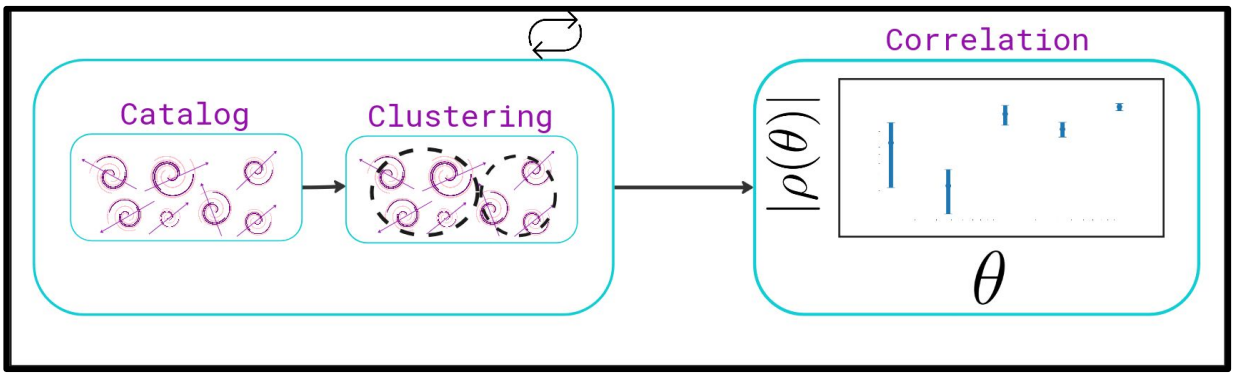
## Contents

# Part I

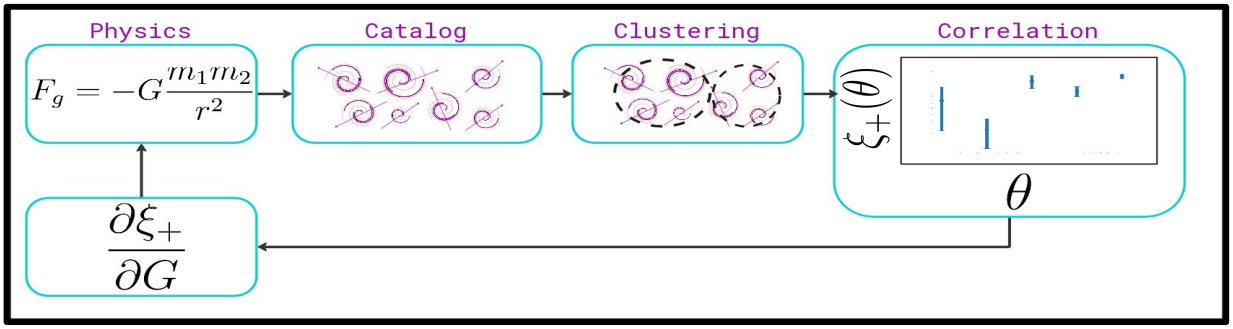
# On Soft Clustering for Correlation Estimators


## Abstract

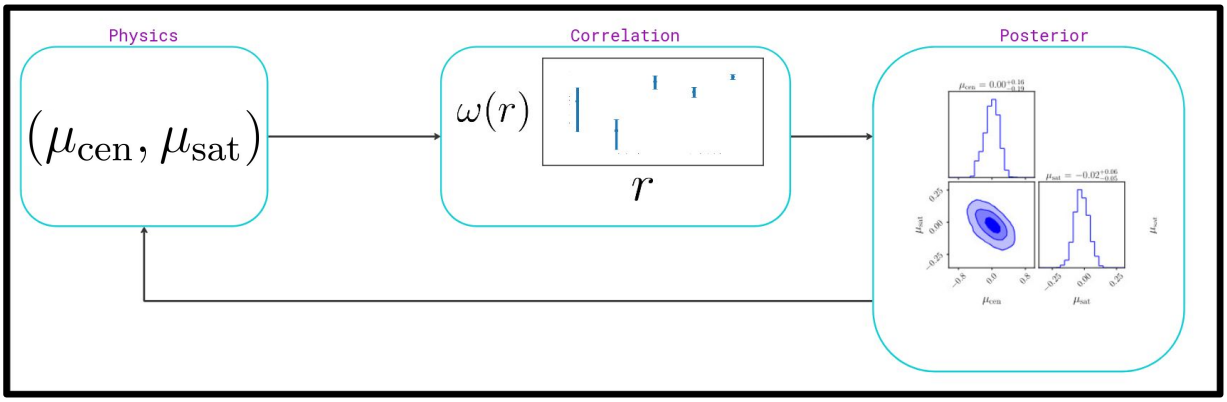
Properly estimating correlations between objects at different spatial scales necessitates $\mathcal{O}(n^2)$ distance calculations. For this reason, most widely adopted packages for estimating correlations use clustering algorithms to approximate local trends. However, methods for quantifying the error introduced by this clustering have been understudied. In response, we present an algorithm for estimating correlations that is probabilistic in the way that it clusters objects, enabling us to quantify the uncertainty caused by clustering simply through model inference. These soft clustering assignments enable correlation estimators that are theoretically differentiable with respect to their input catalogs. Thus, we also build a theoretical framework for differentiable correlation functions and describe their utility in comparison to existing surrogate models. Notably, we find that repeated normalization and distance function calls slow gradient calculations and that sparse Jacobians destabilize precision, pointing towards either approximate or surrogate methods as a necessary solution to exact gradients from correlation functions. To that end, we close with a discussion of surrogate models as proxies for correlation functions. We provide an example that demonstrates the efficacy of surrogate models to enable gradient-based optimization of astrophysical model parameters, successfully minimizing a correlation function output. Our numerical experiments cover science cases across cosmology, from point spread function (PSF) modeling efforts to gravitational simulations to galaxy intrinsic alignment (IA).




(a) An overview of our model uncertainty experiment. We repeatedly cluster using a probabilistic algorithm and look at the resulting standard deviation of the resulting correlations.




(b) An overview of our differentiability experiment. We forward model a gravitational simulation and show that we can differentiate through the estimator even with the clustering approximation.




(c) An overview of our surrogates experiment. We exploit the differentiability of neural networks to enable Hamiltonian Monte Carlo for fast posterior sampling of the IA parameters most likely to minimize a correlation function.

Figure 1: Outline of the three experiments conducted in this paper. The top outlines our model uncertainty experiment (§5), the middle outlines our differentiability experiment (§6), and the bottom outlines our surrogate experiment (§7).

# 1 Introduction

Two- and three-point correlation functions (2/3PCF) are widely used in cosmology. Algorithms to compute these correlation functions first cluster their objects into representative points to ease the computational load of computing $\mathcal{O}(n^2)$ pairwise distances. To numerically compute the correlation function, the pairwise distance between all of the points is then computed, and an estimator is computed on all of the points in each distance bin.

A long-standing goal in the computational sciences is to reconcile epistemic (model) uncertainties from aleatoric (data) uncertainties (Osband et al., 2023; Hüllermeier & Waegeman, 2021). Aleatoric uncertainties describe uncertainty that is inherent to a statistic. This can be due to data quality limitations or inherent stochasticity of the quantity one is trying to model. On the other hand, epistemic uncertainties reflect the inability to generalize from a finite data set due to inherent limitations of the model. In the case of 2PCFs and 3PCFs, the epistemic uncertainty is often overlooked: the best assignment of an object to a given cluster is often unclear, as data points that are "in-between" two or more learned cluster centers can reasonably be chosen to fall in any of the clusters. This ambiguity introduces an epistemic uncertainty, as the model will struggle to appropriately assign objects to representative clusters without enough data. Prior works usually assume this clustering uncertainty to be trivial, arguing that for a significantly large sample, the number of overestimated and underestimated distances are roughly equal, and so the errors tend to cancel out (Jarvis et al., 2004). For example, this was assumed for the Dark Energy Survey (DES) point spread function (PSF) modeling efforts (Jarvis et al., 2021). A PSF is an impulse response of an optical system to light, and the quality of a PSF model is often assessed with $\rho$ statistics (Rowe, 2010; Vogelsberger et al., 2016), which describe correlations in size and shape residuals of a PSF model relative to observed stars. These empirical PSF models are often trained using point source catalogs that are generated by SourceExtractor++ (Bertin & Arnouts, 1996) and filtered to be within a given size/magnitude range. DES covered a vast survey area and included millions of objects which were used to be fit and evaluate PSF models (Dark Energy Survey Collaboration, 2024; Jarvis et al., 2021). With so much data at hand, it was fair to assume that the number of overestimated and underestimated distances were roughly equal, and that downstream modeling errors from clustering were negligible. Surveys such as DES therefore look exclusively at uncertainties quantified via bootstrapping or jackknife (Jarvis et al., 2021; Jarvis & collaborators, 2024), which captures only the model's robustness to individual data points. This is closer in nature to being an aleatoric uncertainty, and we will refer to it as aleatoric uncertainty throughout.

This work presents a method to quantify epistemic uncertainty in regimes where one cannot make the same assumptions as Jarvis et al. (2004) due to the small number of data points available. For instance, clustering errors cannot be assumed to be negligible for the NIRCam PSF modeling efforts with COSMOS-Web (Casey et al., 2023b), where the number of point sources is on the order of a few 100 across the entire field of view (Berman et al., 2024). Even with distance offsets mitigated to a single bin with the TreeCorr algorithm (Jarvis et al., 2004), results can still be heavily biased. A toy example of this with the position-position autocorrelation is presented in Figure 2. The unbiased estimator for this correlation is the Landy–Szalay estimator (Landy & Szalay, 1993) shown in Equation 1.

$$\omega(\theta) = \frac{DD(\theta) - 2DR(\theta) + RR(\theta)}{RR(\theta)} \tag{1}$$

Equation 1 assumes access to one real observation of galaxies as well as an additional catalog of galaxies randomly distributed on the sky. The terms in Equation 1 are $DD(\theta)$, the number of real galaxy pairs separated by an angular distance $\theta$, $RR(\theta)$, the number of random galaxy pairs separated by a distance $\theta$, and $DR(\theta)$, the number of pairs between one real galaxy and one random galaxy separated by a distance $\theta$. Since the estimator is directly proportional to the number of galaxies that are measured to be $\theta$ arcmins apart through $DD(\theta)$, even a single bin offset can heavily skew a measurement. While $\rho$ statistics themselves are not directly related to the number of objects in a single distance bin, the example indicates that binning offsets can constructively bias the result in a given bin.

The core method of this work is to recast object-cluster assignment probabilistically. In this way, we study how the uncertainty related to the clustering of objects can propagate through to the output. Our findings

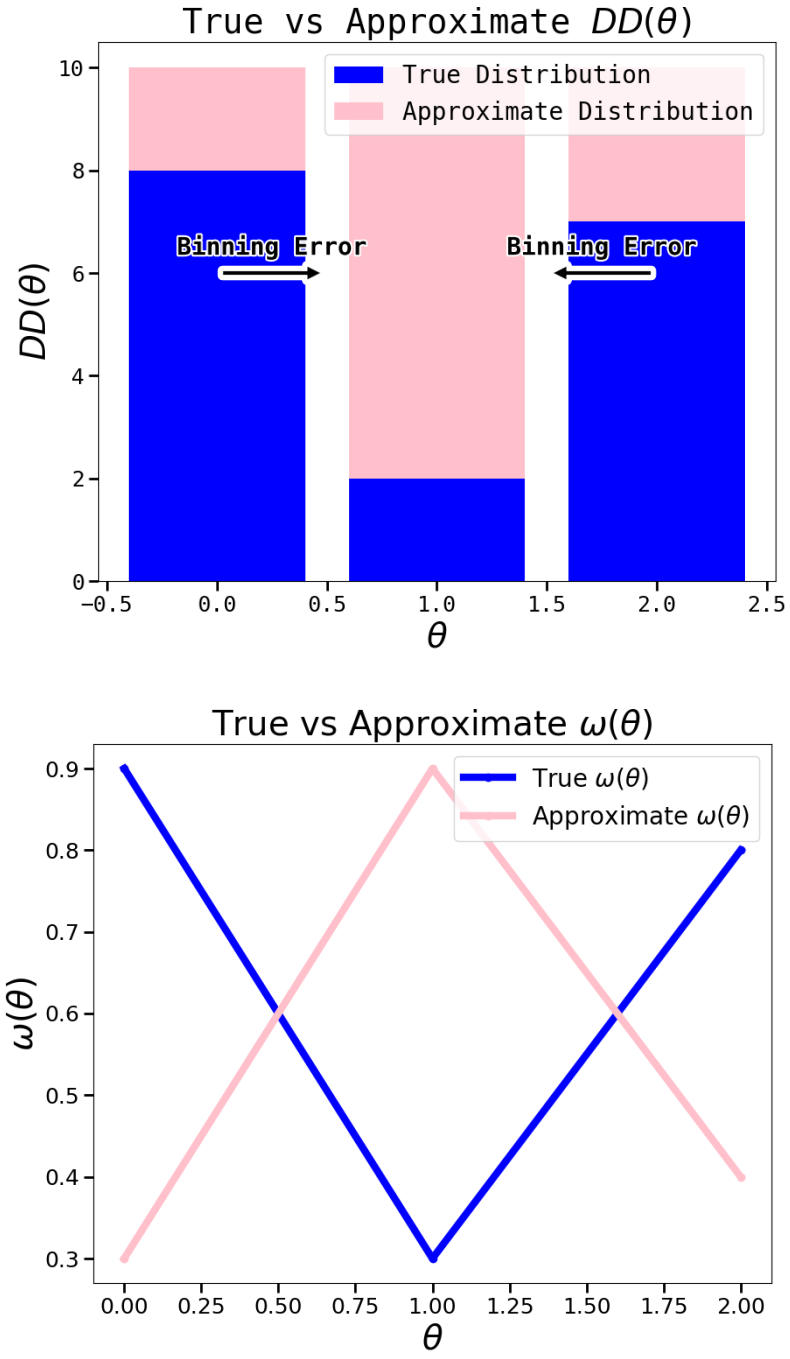



Figure 2: A toy example of a clustering approximation causing a downstream error in the estimation of the $\omega(\theta)$ position-position auto-correlation function.

suggest that there are equal amounts of data and model uncertainty in the case of computing $\rho$ statistics. This further implies that one should skip the clustering step to mitigate the clustering uncertainty.

We also observe that the soft or probabilistic assignment of galaxies to cluster centers makes our estimator theoretically differentiable with respect to its input catalog, and by extension the parameters of any differentiable forward model that produces an input catalog. Differentiability in this context is highly desirable, as it allows one to relate correlations to astrophysical model parameters through gradient-based optimization (for related work, see Lanzieri et al., 2023; Cuesta-Lazaro & Mishra-Sharma, 2024; Lanusse et al., 2023; Campagne et al., 2023; Jagvaral et al., 2024; Berman & McCleary, 2025a; Chantada et al., 2023; Halverson & Pandya, 2024). Halverson & Pandya (2024) in particular describes a pipeline analogous to what we would like to achieve. They optimize cosmological model parameters to maximize the number of extended cosmological epochs where the relative abundances of different $\Omega_i$ remain unchanged, referred to as periods of cosmological stasis (Dienes et al., 2022). In our case, we are interested in what kind of implicit cosmological parameters maximize or minimize different types of two and three-point correlations (cf. Figure 1). Motivated by this desire to maximize or minimize correlation functions, we provide a discussion of the gradient related properties of our algorithm. In contrast to our method, clustering schemes that use hard assignments (i.e. every object has one assignment with unit probability) often depend on functions like MEDIAN or ARGMIN, which results in derivatives that are either identically zero or ill-defined. We further explore the differentiability of our clustering scheme with an experiment. In our experiment, we use a gravitational simulation as a forward model with the goal of differentiating a correlation function with respect to the gravitational constant, $G$. Our findings motivate the use of surrogate models: learned functions that approximate the relationship between astrophysical model parameters and observable data. Recent studies demonstrate their ability to directly model two-point statistics from astrophysical model parameters (Pandya et al., 2024; 2025b), making them ideal for our purpose. Accordingly, we extend the work of Pandya et al. (2024; 2025b), using the Intrinsic Alignment Emulator (IAEmu) surrogate model to optimize seven halo-based modeling parameters to minimize a correlation function. In doing so, we illustrate the effectiveness of surrogate models to enable gradient-based optimization of astrophysical model parameters, which in turn allows us to study what physical circumstances cause correlations to be maximized or minimized.

Our paper executes three experiments that are designed to explore each of the three themes outlined in our title: model uncertainty, differentiability, and surrogates.

1. **Model Uncertainty and PSFs (§5):** For this experiment, we use ShOpt.jl (Berman et al., 2024; Berman & McCleary, 2024a) to produce PSF models for the COSMOS-Web NIRCam images (Casey et al., 2023b). Using our proposed algorithm, we then compute $\rho$ statistics. Finally, we assess the epistemic uncertainty associated with our model and compare it to the uncertainty we would get with traditional bootstrapping techniques. Our PSF modeling experiment is geared towards observers, who do not have access to differentiable forward models of PSF residuals.

2. **Differentiability and Gravity Simulations (§6):** For this experiment, we forward model gravitational interactions using a series of ordinary differential equations. We then compute correlation functions on the resulting catalog and test different techniques for differentiating the correlation function outputs with respect to the cosmological parameters that determined the simulation. Our gravitational simulation experiment is designed with theorists in mind. Theorists can produce multiple realizations of their model through simulations that are derived from different cosmological parameters. Here, we assess the feasibility of differentiating a correlation function with respect to $G$.

3. **Surrogates and Galaxy Intrinsic Alignment (§7):** For this experiment, we use an existing surrogate model to relate astrophysical model parameters to IA (Pandya et al., 2024; 2025b). Using this surrogate, we use Hamiltonian Monte Carlo (HMC) to find posterior distributions over astrophysical model parameters that are likely to correspond to correlations of zero. Our IA experiment shows how to compensate for forward models and estimators that are not easily differentiated, and are again geared towards theorists who are able to realize many instances of their forward model through inference in order to train a surrogate. Our usage of HMC leverages the differentiability of

neural networks to accelerate the sampling procedure in comparison to traditional algorithms like Metropolis–Hastings (Betancourt, 2017; Robert et al., 2004).

The remainder of the paper is organized as follows: §2 discusses the relevant notation and theory for $\rho$ statistics and IA, §3 follows with an outline of the data products used in our study, §4 lays out additional notation as well as details for the algorithm used in this paper, and §5 -7 presents the results of the experiments described above followed by our conclusions in §8.

## 2 Notation and Theory

In this section we discuss the relevant theory and notation for correlations of PSF size and shape residuals and of IA. We also expand on the theory we use to compute $\sigma_{\text{epistemic}}$ and $\sigma_{\text{aleatoric}}$.

### 2.1 PSF Modeling

A PSF describes the impulse response of an optical system to light. Effects like diffraction, optical aberrations, atmospheric turbulence (if applicable), and telescope jitter are summarized in the telescope's PSF. For the James Webb Space Telescope (JWST), the PSF provides an obstacle for science goals including the highest resolution dark matter mass maps (Scognamiglio, 2024, Scognamiglio et al., in prep) and the characterization of over 100 strong lens systems (Nightingale et al., in prep). $\rho$ statistics are a suite of 2PCFs introduced in Rowe (2010) and expanded upon in Vogelsberger et al. (2016) for characterizing biases in size and shape measurement that arise from PSF modeling.

We adopt the same notation for describing $\rho$ statistics as Jarvis et al. (2021) and McCleary et al. (2023a):

$$\rho_1(\theta) \equiv \langle \delta e^*_{\text{PSF}}(\boldsymbol{x}) \delta e_{\text{PSF}}(\boldsymbol{x}+\boldsymbol{\theta}) \rangle \tag{2}$$

$$\rho_2(\theta) \equiv \langle e^*_{\text{PSF}}(\boldsymbol{x}) \delta e_{\text{PSF}}(\boldsymbol{x}+\boldsymbol{\theta}) \rangle \tag{3}$$

$$\rho_3(\theta) \equiv \left\langle \left( e^*_{\text{PSF}} \frac{\delta T_{\text{PSF}}}{T_{\text{PSF}}} \right)(\boldsymbol{x}) \left( e_{\text{PSF}} \frac{\delta T_{\text{PSF}}}{T_{\text{PSF}}} \right)(\boldsymbol{x}+\boldsymbol{\theta}) \right\rangle \tag{4}$$

$$\rho_4(\theta) \equiv \left\langle \delta e^*_{\text{PSF}}(\boldsymbol{x}) \left( e_{\text{PSF}} \frac{\delta T_{\text{PSF}}}{T_{\text{PSF}}} \right)(\boldsymbol{x}+\boldsymbol{\theta}) \right\rangle \tag{5}$$

$$\rho_5(\theta) \equiv \left\langle e^*_{\text{PSF}}(\boldsymbol{x}) \left( e_{\text{PSF}} \frac{\delta T_{\text{PSF}}}{T_{\text{PSF}}} \right)(\boldsymbol{x}+\boldsymbol{\theta}) \right\rangle \tag{6}$$

where $e_{\text{PSF}}$ is the ellipticity of the real PSF, i.e., the star ellipticity, $T_{\text{PSF}}$ is the size of the real PSF, $\delta e_{\text{PSF}}$ is the difference between the ellipticity of the real and model PSFs at position $\boldsymbol{x}$, and $\delta T_{\text{PSF}}$ is the difference between the sizes of the real and model PSFs at position $\boldsymbol{x}$. Brackets denote averages over all pairs within a separation $\boldsymbol{\theta}$, and asterisks denote complex conjugates.

$\rho$ statistics specify the $\xi_+$ weak lensing shear correlations, which are given by

$$\xi_+(\theta) = \langle \epsilon_t(x+\theta)\epsilon_t(x) \rangle + \langle \epsilon_\times(x+\theta)\epsilon_\times(x) \rangle \tag{7}$$

and estimated by

$$\hat{\xi}_+(\theta) = \frac{\sum_{i,j} w_i w_j (\epsilon_{t,i}\epsilon_{t,j} + \epsilon_{\times,i}\epsilon_{\times,j})}{\sum_{i,j} w_i w_j}. \tag{8}$$

where $w_{i/j}$ represents the weight uncertainties of object $(i/j)$, $\epsilon_{t,i/j}$ represents tangential shear of object $(i/j)$ and $\epsilon_{\times,i/j}$ represents the cross shear of object $(i/j)$ (Kilbinger, 2015; Schneider et al., 2002). The complex components of the products in Equations 2 - 6 are known to average out to zero due to parity symmetry (Rowe, 2010; Schneider et al., 2006), and are thus ignored in the estimator.

### 2.2 Galaxy Intrinsic Alignment

Intrinsic alignment describe correlations between galaxy orientations and the underlying distribution of dark matter where they are embedded; see Lamman et al. (2023) for a review. The three relevant IA correlation functions are the galaxy position-position $\xi(r)$, position-orientation $\omega(r)$, and orientation-orientation $\eta(r)$ correlation functions, and are estimated according to Equations 9-11.

$$\xi(r) = \left\langle \frac{n(r) - \bar{n}(r)}{\bar{n}(r)} \right\rangle \tag{9}$$

$$\omega(r) = \langle |\hat{e}(\mathbf{x}) \cdot \hat{r}|^2 \rangle - \frac{1}{3} \tag{10}$$

$$\eta(r) = \langle |\hat{e}(\mathbf{x}) \cdot \hat{e}(\mathbf{x} + \mathbf{r})|^2 \rangle - \frac{1}{3} \tag{11}$$

where $n(r)$ is the number of galaxies separated by distance $r$, $\bar{n}(r)$ is the expected number of galaxies separated by distance $r$ for a random distribution, $x$ is the position vector of a galaxy, and $\hat{e}(x)$ is a 3D orientation unit vector of a galaxy.

### 2.3 Uncertainty Quantification

Given a probabilistic clustering algorithm for estimating correlations, $\sigma_{\text{epistemic}}$ is computed by estimating the correlation value in each distance bin $n$ times and then computing the standard deviation of the distribution. For computing $\sigma_{\text{aleatoric}}$ via bootstrapping or jackknife, one must first make the probabilistic clustering algorithm deterministic. In this work, we use greedy sampling to accomplish this; that is, each object is always assigned to the cluster with its highest membership probability. In this way, the model uncertainty is isolated from the data uncertainty. To perform bootstrapping, random subsets of the input data is removed over $n$ iterations and the correlation value is estimated in each distance bin. $\sigma_{\text{aleatoric}}$ is then computed via the standard deviation of the output in each distance bin. We choose $n = 10$ samples to obtain the error bars.

Our definitions of aleatoric and epistemic are consistent with what is used in the machine learning literature (Hüllermeier & Waegeman, 2021). It is worth noting that in the cosmology literature, these uncertainties are sometimes described as statistical and systematics, e.g. in Freedman et al. (2024). Following Freedman et al. (2024), we take our uncertainties to be additive. That is, correlation functions will have some value $\mu \pm \sigma_{\text{aleatoric}} \pm \sigma_{\text{epistemic}}$ in each distance bin. $\mu$ can be obtained from the average of the either the probabilistic or deterministic algorithm samples.

## 3 Data

### 3.1 Point Source Catalog

The point sources used for generating our PSF models come from the SE++ (Bertin & Arnouts, 1996) COSMOS-Web catalog (Casey et al., 2023b, Shuntov et al. in prep). We restrict ourselves to imaging from the F115W NIRCam (Rieke et al., 2003; 2005; Beichman et al., 2012) wavelength filter, as the PSFs in this wavelength are the closest to being well-approximated by a Gaussian (Berman et al., 2024). $\rho$ statistics and the related Hirata–Seljak–Mandelbaum (HSM) adaptive moments (Hirata & Seljak, 2003; Mandelbaum et al., 2005) are based off of the Gaussian approximation of the PSF, and so the use of $\rho$ statistics in our experiments is most appropriate in this wavelength. PSF models are obtained using ShOpt.jl (Berman et al., 2024; Berman & McCleary, 2024a). Our choice of PSF fitter is discussed in more detail in Appendix A. Our ShOpt fits can be seen in Figure 3. The ellipticites of the objects in our data set are also visualized in Figure 4. We use the same configuration file as found in the appendix of Berman et al. (2024). Our data set statistics for PSF modeling are summarized in Table 1.

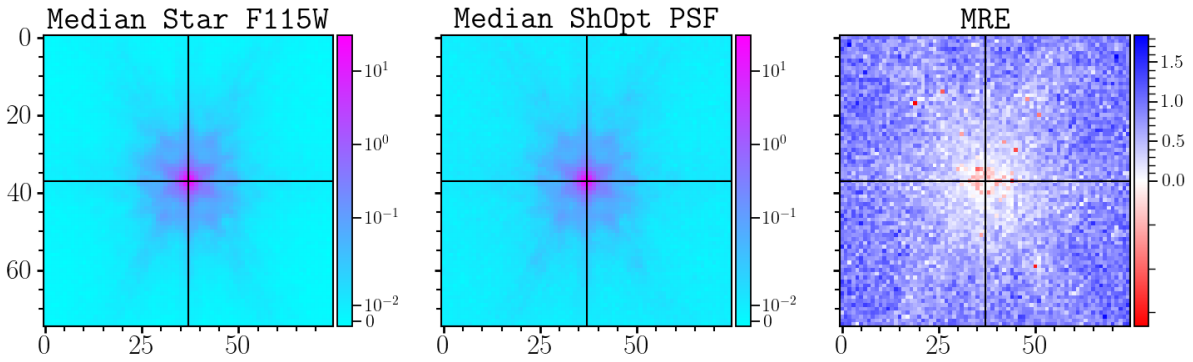



Figure 3: ShOpt PSF fit used in this work. The left panel shows the median of the vignettes. The center panel show the median PSF cutout. The right panel show the average relative error between the vignette cutouts and the PSF cutouts. More on these plots can be found in Berman et al. (2024).

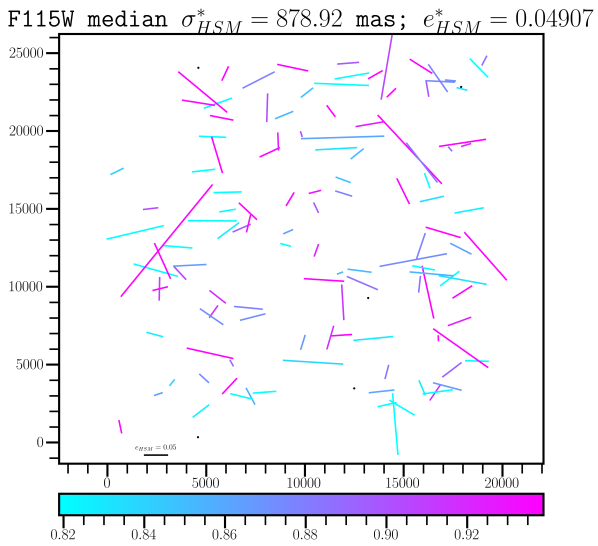



Figure 4: A quiverplot indicating the sizes and shape of the point sources computed via HSM adaptive moments in a single tile of the COSMOS-Web survey. The color bar at the bottom denotes the size of the source, the length of the quiver describes the magnitude of ellipticity, and the orientation and position describes its measured orientation and position on the image plane. The median size and ellipticity of point sources in terms of HSM moments is given at the top of the figure.

| Training Stars | Validation Stars | Filter | Survey Area ($\deg^2$) |
|---|---|---|---|
| 6333 | 610 | F115W | 0.59 |

Table 1: Data set statistics for PSF catalog

### 3.2 Simulated Galaxy Catalog

There are plenty of widely-used physically realistic N-body simulations, using publically available codes such as Gadget-2/3 (Springel, 2005), ART (Kravtsov et al., 1997), RAMSES (Teyssier, 2002), and Enzo (Brummel-Smith et al., 2019; Bryan et al., 2014), see Vogelsberger et al. (2020) for a review. However, this work aims only to have a simple differentiable forward model. Instead of using Gadget-2 or similar tools, we write our simulations from scratch in Julia and Python and time evolve objects according to ordinary differential equations from Newtonian mechanics. Our main implementation uses Jax (Bradbury et al., 2018) with Diffrax (Kidger, 2021). We specify starting positions and velocities for objects $[\vec{x}, \vec{v}]$, which are then evolved via $\frac{d\vec{x}}{dt} = \vec{v}$, $\frac{d\vec{v}}{dt} = \vec{a}$, and $\vec{a} = -\frac{GM}{|\vec{x}|^3}\vec{x}$. We begin by randomly initializing galaxies with some mass $M$, velocity $\vec{v}$, and initial position $\vec{r}$. These galaxies exist in a gravitational field where the galaxy-galaxy gravitational attraction is assumed to be negligible compared to the attraction toward the center of the field. Thus, we can solve each ODE system in Diffrax independently and use the vmap function to efficiently time evolve each object. While our toy simulation is not very physically meaningful, this approach outputs catalogs that are differentiable with respect to the input cosmology, which in this case is just the gravitational constant $G$. The number of galaxies in the simulation can vary, allowing us to study the efficacy of our algorithm as a function of catalog sizes.

### 3.3 Intrinsic Alignment Catalog

This work uses an extensive galaxy catalog that was generated according to seven halo-based modeling parameters using the procedure of Pandya et al. (2024; 2025b). The halo occupation distribution (HOD) model is used to populate existing catalogs of dark matter-only halos with galaxies (Hearin et al., 2017). The model is parameterized by five occupation components, $\log M_{\min}$, $\sigma_{\log M}$, $\log M_0$, $\log M_1$, and $\alpha$, described in detail in (Zheng et al., 2007). Briefly, the occupation components parameterize the mean occupation functions given by

$$\langle N_{\text{cen}}(M)\rangle = \frac{1}{2}\left[1 + \text{erf}\left(\frac{\log M - \log M_{\min}}{\sigma_{\log M}}\right)\right] \tag{12}$$

and

$$\langle N_{\text{sat}}(M)\rangle = \frac{1}{2}\left[1 + \text{erf}\left(\frac{\log M - \log M_{\min}}{\sigma_{\log M}}\right)\right]\left(\frac{M - M_0}{M_1'}\right)^{\alpha} \tag{13}$$

where

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt. \tag{14}$$

The contributions of Van Alfen et al. (2024) ensures that the model takes into account IA, including the central alignment strength $\mu_{\text{cen}}$ and the satellite alignment strength $\mu_{\text{sat}}$.

For each of the seven input parameters, ten realizations of the model were produced. Once the catalogs were generated, the three IA correlation functions were estimated according to Equations 9 - 11. This led to a data matrix of the shape $(110,526; 7; 100; 3; 20)$. In other words, we have $110,526$ tuples of seven input values that produced ten realizations of three correlation functions that each occupy twenty radial bins. This catalog is used to train the IA emulator (`IAEmu`) surrogate model introduced in Pandya et al. (2024). `IAEmu` is a neural network based emulator designed to predict galaxy IA statistics from a given HOD, with the alignments parameterized according to the two parameter family $\mu_{cen}$ and $\mu_{sat}$ (Van Alfen et al., 2024). These $110,526$ tuples are broken down in training, validation, and testing sets for IAEmu. This is summarized in Table 2.

| Training Set | Validation Set | Testing Set | Input Parameters | Output Bins | Realizations |
|---|---|---|---|---|---|
| 77,368 (70%) | 11,052 (10%) | 22,105 (20%) | 7 | 20 | 10 |

Table 2: Data set statistics for the IA catalogs

## 4 Algorithm Overview

In this section, we outline our algorithm used to estimate correlations. Our algorithm has two variants, one geared toward estimating clustering uncertainties and the other towards differentiability. Two variants are needed because estimating cluster uncertainty requires sampling, which is incompatible with differentiation. There are a variety of ways to adapt sampling to be a differentiable process, which we explore in depth in §6. Throughout this work, we use most of the formalism laid out in Bezdek et al. (1984). Our formalism differs in that it uses a distance function to compute angular distances on the sky rather than using a matrix to define a norm. We also define the quantity matrix, which allows us incorporate the quantities we wish to correlate into the original formalism. For details on the fuzzy sets formalism, see Zadeh (1965) and (page 38, Aluffi, 2021). Algorithm 1 first uses K-means++ (Kapoor & Singhal, 2017) to initialize a series of cluster centers. This algorithm adds initial cluster centers sequentially by considering the distances to centers that have already been determined. This initialization has been shown to help clustering algorithms and is better than choosing initial cluster centers at random (Kapoor & Singhal, 2017). We next use the fuzzy-c-means algorithm (Bezdek et al., 1984) to obtain final cluster centers and weight assignments that minimize the functional

$$J_m(U, v) = \sum_{k=1}^{N} \sum_{i=1}^{c} (U_{ik})^m \|y_k - v_i\|^2 \tag{15}$$

where

- $Y = \{y_1, \ldots, y_n\}$ are the object positions,
- $c$ is the number of clusters in $Y$; $2 \leq c < n$,
- $m$ is the weighting exponent (or fuzziness); $1 \leq m < \infty$,
- $U$ is the fuzzy c-partition of $Y$; $U \in \mathbb{R}^{c \times N}$
- $v = (v_1, \ldots v_c)$ is a vector of centers,
- $v_i = (v_{i1}, \ldots, v_{in})$, is the center of cluster $i$.

We also define $q_{ij}$ with $1 \leq i \leq M$ and $1 \leq j \leq N$ as the quantity matrix. Each row represents a quantity of interest that we wish to correlate and each column represents a different object. In this work, there will be two rows corresponding to two shears. Our formalism differs from Bezdek et al. (1984) in that the norm on $\|y_k - v_i\|^2$ is not induced by a matrix. Instead, we use the Vincenty formula given by Equation 16 to compute angular separations on the sky between two angular coordinates $(\phi_1, \lambda_1), (\phi_2, \lambda_2)$:

$$\Delta\sigma = \arctan\left(\sqrt{(\cos\phi_2 \cdot \sin\Delta\lambda)^2 + (\cos\phi_1 \cdot \sin\phi_2 - \sin\phi_1 \cdot \cos\phi_2 \cdot \cos\Delta\lambda)^2}, \sin\phi_1 \cdot \sin\phi_2 + \cos\phi_1 \cdot \cos\phi_2 \cdot \cos\Delta\lambda\right). \tag{16}$$

While this does not have a compact matrix representation, it is stable at all numerical scales (Vincenty, 1975). In this way, we are sacrificing speed for stability. This design choice is discussed more in §6. The two argument arctan function is used to constrain the solution to the correct quadrant. Our usage of the Vincenty formula also assures that our distance estimates are consistent with Astropy (Robitaille et al., 2013), which also uses the Vincenty formula for this task.

Our implementation of fuzzy-c-means works as follows:

1. Initialize centers via K-means++ and weights via random initialization.

2. Update centers via

$$\hat{v}_i = \frac{\sum_{k=1}^{N} \left(\hat{U}_{ik}\right)^m y_k}{\sum_{k=1}^{N} \left(\hat{U}_{ik}\right)^m}; 1 \leq i \leq c \tag{17}$$

3. Update weights via

$$\hat{U}_{ik} = \left( \sum_{j=1}^{c} \left( \frac{\hat{d}_{ik}}{\hat{d}_{jk}} \right)^{\frac{2}{m-1}} \right)^{-1} \tag{18}$$

where the distances $d$ are given by Equation 16.

4. Repeat steps 2 and 3 until either the maximum number of iterations is reached or the difference in weights between iterations is less than a user-provided tolerance and matrix norm.

After obtaining final centers $v$ and weights $U$, we can assign each galaxy to a cluster center via sampling. Each row in $U^T$ represents the normalized probability distribution that an object belongs to a cluster with center $v_i$. Thus, each object is assigned to a cluster via

$$M_i \sim \text{Categorical}(U_i^T). \tag{19}$$

We define the quantities of each cluster as the average of all the quantities of the objects assigned to it. Note that this is unnecessary if there are no quantities. This may arise if one is computing a position-position autocorrelation or similar. From here, we can proceed by computing all of the pairwise distances between clusters and then use a known estimator to compute the correlation in each distance bin. The whole of Algorithm 1 is recapitulated in the algorithm block below.

**Algorithm 1** Probabilistic Estimation

1: Initialize centers $v$ using K-Means++
2: Compute new centers $v$ and weights $U$ using fuzzy-c-means
3: Assign objects to clusters via $M_i \sim \text{Categorical}(U_i)$
4: Define new objects with centers $v$ and quantities given by the average of all objects in the cluster
5: Compute all pairwise distances $d$ between objects using Equation 16
6: Bin the pairwise distances by spatial separation
7: **for** each bin $b_i$ **do**
8: Evaluate correlation estimator on binned object distance pairs
9: **end for**=0

Algorithm 2 is a differentiable analog to algorithm 1. Algorithm 1 goes from soft bin assignments to hard assignments by sampling the rows of $U$. By soft assignments, we mean that for each object the assignment probabilities are spread across the different clusters. This is in contrast to our definition of a hard assignment in §23. This transition from hard assignment to soft assignments is the first obstacle to be amended in Algorithm 2, since sampling is not a differentiable process. There are at least three ways to adapt this step in Algorithm 1 to become differentiable, which we will hereafter refer to as approaches a, b, and c:

[label=()]Use the Gumbel Max reparameterization trick to enable differentiation through the sampling process (Maddison et al., 2016; Jang et al., 2016) [1]. Avoid sampling entirely and instead compute weighted averages to get the quantities for each cluster. The quantities are weighted by how likely they are to belong to an object in a given cluster and then averaged. This is done for each cluster. Assign each object to a cluster with the highest membership probability via ARGMAX

[1] An example implementation of this can be found here: https://github.com/cassanof/gumbel-bucket-rs.

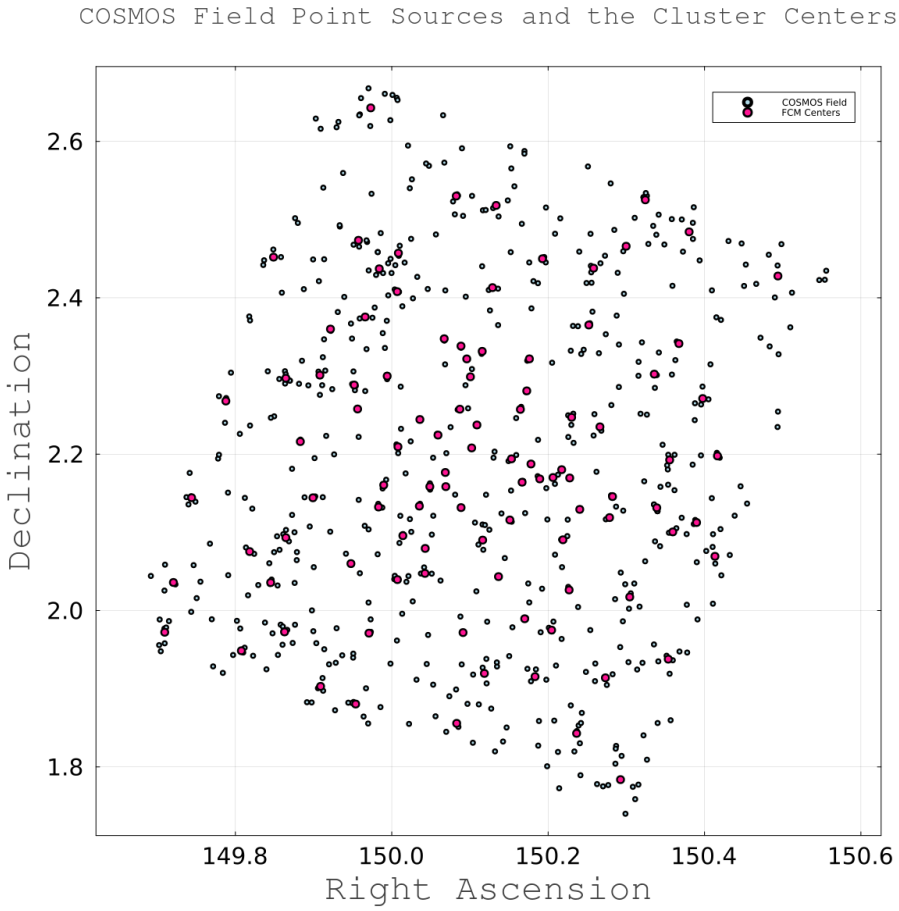



Figure 5: The cluster centroids found via fuzzy-c-means overlayed on the point sources of the underlying COSMOS field

(effectively proceeding with hard assignments). Adapt the membership matrix such that there is a 1 in the entries $U_{ij}$ where object $j$ most likely belongs to cluster $i$. Then, make new objects by averaging the quantities in each cluster. During the gradient calculation, approximate the gradients for each object as the gradient of the new cluster object it belongs to. In other words, if object $i$ is in cluster $j$, approximate the gradient of a correlation function with respect to object $i$ as the gradient of the correlation function with respect to the new object generated from cluster $j$.

We explore approaches $b$ and $c$ because they are simple to compute via matrix multiplication. For these approaches to work, we must also normalize the rows of $U$. We define

$$\tilde{U}_{ij} = \frac{U_{ij}^T}{\sum_{k=1}^{m} U_{kj}^T}. \tag{20}$$

Now we can express our weighted average with the matrix product

$$\tilde{q} = q\tilde{U}. \tag{21}$$

We see that $\tilde{q} \in \mathbb{R}^{M \times c}$, representing the $M$ quantities for each cluster $c$. Typical correlations choose $M$ to be 2 or 3. For example, if we were computing the $\xi_+$ shear-shear correlation, we would have two quantities $e$ and $e^*$ for each cluster.

Another adaptation we need to make from algorithm 1 is the hard assignment of distance pairs to bins. Algorithm 1 takes all of the distance pairs that fall into a distance bin and runs an estimator to get the

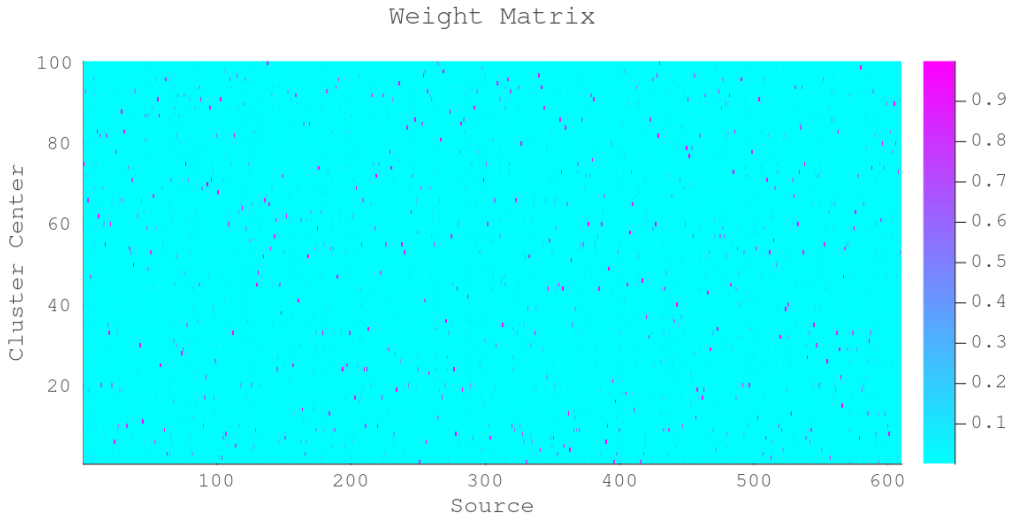



Figure 6: An example weight matrix $U$ produced via fuzzy-c-means for 610 point sources in the COSMOS field and 100 cluster centers.

correlation value for that bin. The differentiable analog is to compute the estimator over *all* distance pairs, where the contribution of each distance pair is weighted by how likely it is to fall into a given bin with bin edges $a$ and $b$. In the case of the shear-shear correlation, this extends naturally the known estimator in Equation 8. We determine the weights $w_i w_j$ using the product of two sigmoid functions, which gives high importance to distances within the bin and almost no importance to objects that are outside of it. We explore two weighting functions. The first is given by

$$W(a, x, b) = \frac{1}{1 + e^{-\alpha(x-a)}} \cdot \frac{1}{1 + e^{\alpha(x-b)}} \tag{22}$$

and the second by

$$W(a, x, b) = e^{-\alpha\left(x - \frac{b-a}{2}\right)^2}. \tag{23}$$

Both weighting functions exploit the sharp decline of the exponential function to quickly decrease the weights. An example of this for the first weighting function is shown in Figure 7 for distances between 1 and 2. Algorithm 2 is summarized in the algorithm block below. Given that both weighting functions can have their decay controlled by the $\alpha$ parameter, we found that both weighting functions can be tuned to produce correlation outputs consistent with other packages used for estimating correlation functions.

**Algorithm 2** Differentiable Estimation

1: Initialize centers $v$ using K-Means++
2: Compute new centers $v$ and weights $U$ using fuzzy-c-means
3: Define new objects with centers $v$ and quantities $q\tilde{U}$
4: Compute all pairwise distances $d$ between objects using Equation 16
5: Compute the probability that each object pair falls in distance bin $b_i$ by weighting by the sigmoid function in Equation 22 or the Gaussian function in Equation 23
6: **for** each bin $b_i$ **do**
7: Evaluate correlation estimator on galaxy pairs weighted by their probability of being in bin $b_i$
8: **end for**=0

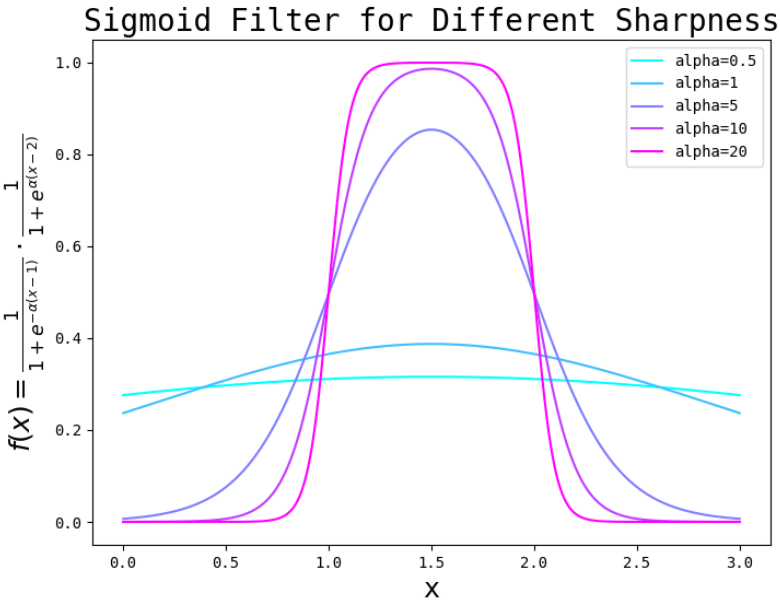



Figure 7: An example weighting for estimates for a distance bin bounded below by 1 and above by 2 for various sharpnesses.

## 5 Model Uncertainty

In this section, we discuss the results of Algorithm 1 on our dataset, which we compare with a bootstrapping baseline to obtain epistemic and aleatoric uncertainties.

Figure 5 shows the cluster centers overlayed with the objects on the sky. The result indicates that the cluster centers reasonably approximate the spatial variation of the sources in the underlying COSMOS field. Figure 6 shows the weight matrix after the objects are clustered with fuzzy-c-means.

Figure 9 shows the Shannon information entropy and maximum probability of each row in the weight matrix. While the majority of objects have a clear preference toward being assigned to a specific cluster, there is a non-trivial number of objects that could be reasonably assigned to multiple clusters. This indicates that the model is often not sure where the best assignment of an object to a cluster is. For this reason, we should expect the error bar associated with model uncertainty to be comparable to data uncertainty. This bears out in Figure 8 and Figures 44 - 47.

We note that increasing or decreasing the number of clusters did not significantly reduce the size of the model uncertainty. This indicates that the source density in our COSMOS-Web data set is not large enough to beat down the errors associated with clustering. Recall that in §2 we said that we are treating uncertainties as additive. Figures 8 and 44 - 47 therefore indicate that in many cases the epistemic uncertainties doubles the uncertainty compared to bootstrapping alone. This is especially true for the $\rho_1$, $\rho_2$, and $\rho_4$ statistics. For this reason, we suggest a "naïve"[2] approach to computing $\rho$ statistics, wherein no clusters are used and the correlation estimators are used directly on the catalogs. Given that we have on the order of $\sim 500$ objects, the number of pairwise distances is not too large to store and the number of distance calculations to compute is not prohibitively expensive.

[2]The use of the word naïve is in reference to its usage in Jarvis et al. (2004).

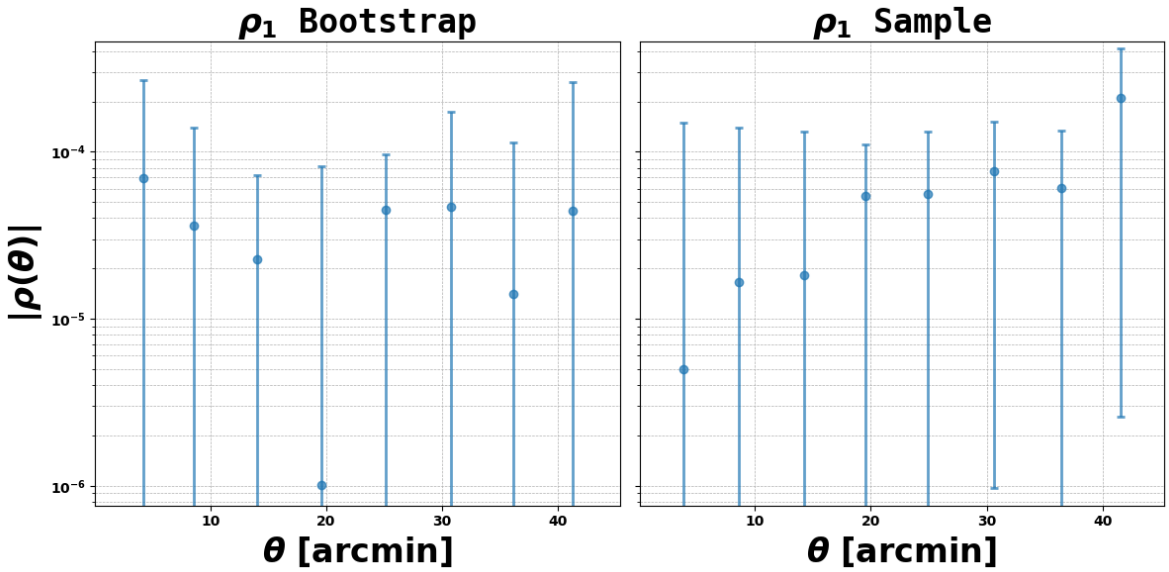



Figure 8: $\rho_1$ correlation function with data taken from the COSMOS-Web point source catalogs. Error bars drawn from bootstrap (left) and sampling (right) approaches. Negative correlations are shown in absolute value, and correlations are plotted in logarithmic scale.

## 6 Differentiability

In this section we outline the utility of algorithm 2 for gradient-based optimization. To do this, we create a simulated catalog of galaxies moving in a gravitational field. We time evolve the objects until some terminus $T_{\max}$, which was found by considering an object undergoing uniform circular motion and calculating how long it would take to complete one orbit. For simplicity, we analyze correlations across three linearly spaced angular separation bins. A max separation of 160 arcmins is chosen to ensure each of the three bins have enough samples. We seek to differentiate the shear-shear correlation $\xi_+(\theta)$ with respect to the gravitational constant $G$ at each angular separation bin $\theta_i$, illustrating an example of enabling differentiability of a correlation with respect to an astrophysical model parameter. The ellipticities are predetermined, but the final positions are naturally dependent on the underlying physics. Since the catalog is produced by solving a series of ordinary differential equations (ODEs), computing the derivative just amounts to using the chain rule $\frac{\partial \xi}{\partial G} = \sum_{i=1}^{N} \frac{\partial \xi}{\partial \alpha_i} \frac{\partial \alpha_i}{\partial G} + \frac{\partial \xi}{\partial \delta_i} \frac{\partial \delta_i}{\partial G}$. Here, $\alpha$ denotes the right ascension and $\delta$ denotes the declination coordinates. We use the ODE solvers and automatic differentiation tools implemented in Julia (Bezanson et al., 2017). Specifically, we use the OrdinaryDifferentialEquations.jl (Rackauckas & Nie, 2017a) library for working with differential equations. We also tested both reverse mode automatic differentiation with Zygote.jl (Innes et al., 2019) and forward mode automatic differentiation with ForwardDiff.jl (**?**). The latter proved to be more effective. We also built an implementation with Jax (Bradbury et al., 2018) and Diffrax (Kidger, 2021), which is available on our GitHub repository. Our adoption of these tools is motivated by the discussion of Berman & Ginesin (2024).

For both approaches b and c (as defined in §4), the naive approach to computing the gradient $\frac{\partial \xi}{\partial \alpha_i}$ or $\frac{\partial \xi}{\partial \delta_i}$ would involve tracking the computation graph through the entire fuzzy-c-means algorithm, which can sometimes take over 100 iterations. Another complication is that the distance calculations and normalization computations cannot be expressed through matrix multiplication, making it difficult to accelerate the gradient calculations with a GPU. For approach $b$, taking the gradients in this manner proved to be prohibitively expensive via reverse mode automatic differentiation, for the reasons outlined above. With forward mode differentiation, the gradient calculation was on the order of a minute. This is expected, since we are differentiating several output bins with respect to 1 input (gravity).

One workaround to the computation time bottleneck is to pre-compute the final centers and weights with fuzzy-c-means, then update the weights and centers for only a single iteration. This makes the final clustering a function of the data without keeping the entire history of the clustering procedure in the computation graph. Using this "trick" to approximate the true gradient, we can roughly halve the computation time needed to solve for the derivative $\frac{\partial \xi}{\partial G}$. However, when using approach $b$, issues related to precision still exist. When using a large number of clusters, we found that the Jacobian relating the dependence of weights on the input data was sparse. The sparsity result is expected, as local perturbations of input positions, by design, preferentially impact the assignment probabilities of nearby cluster centers. As a result, we are left with vanishing gradients, and the algorithm becomes unstable as we scale the number of clusters, producing `NaN` outputs. We note that these `NaN` issues did not occur on randomly computed inputs used for testing the differentiability of each individual function, indicating that the reliability of this method will vary for problems of different conditioning. This also confirms that while each individual step in the algorithm is differentiable, instabilities can still occur from gradient multiplication. It is worth noting that multiplying a matrix with another matrix that is sparse does not mean the resulting matrix is necessarily sparse. Thus, one Jacobian *alone* being sparse is not the problem, rather, its the effect of a sparse Jacobian when propagated through the chain rule. Because we are reasoning about gradients, we want to ensure our representations are dense. We note that this is in contrast to other numerical problems where sparsity is necessary to maintain computational efficiency when dealing with large matrices (chapter 10 of Saad, 2003; Rosen, 2022).

Approach $c$ is able to circumnavigate some of these issues. As evidenced by Figure 10, we are successfully able to differentiate the outputs of our correlation function in each distance bin with respect to the input astrophysical model parameter $G$ at the point $G = 1$. However, issues with scaling the number of data points still persist, as the computation becomes too large to run on a CPU well before 500 galaxies are simulated via the forward model.

The precision and computation time issues suggest that the simple solution is not always sufficient for enabling the differentiability of correlation functions with respect to the underlying model inputs. Surrogate models, however, have demonstrated the ability to succesfully emulate correlation function estimators (Van Alfen et al., 2024; Pandya et al., 2024; 2025b). These surrogate models can be as simple as linear regression, or more involved functions such as those represented by a neural network (NN). NNs, by definition, are differentiable models (i.e. trained with gradient descent) and enjoy GPU accelerated computations.

## 7 Surrogates

In this section, we utilize the surrogate model developed in Pandya et al. (2024; 2025b), `IAEmu`, which estimates galaxy IA correlations from an underlying set of HOD parameters. We use `IAEmu` to optimize model parameters that minimize the IA correlations, to study regions of parameter space that correspond to minimal IA contamination and are thus of physical interest. `IAEmu` serves as an emulator for Halotools-IA, which extends traditional HOD modeling to include IA information through a two parameter family, $\mu_{\rm cen}$ and $\mu_{\rm sat}$, which govern central and satellite alignment strengths, respectively. More specifically, `IAEmu` outputs the galaxy position-position ($\xi$), position-orientation ($\omega$), and orientation-orientation ($\eta$) correlations as well as estimates of the aleatoric uncertainty (shape noise) of each correlation. This noise is not physical, and corresponds to variance across realizations of the underlying HOD due to limited cosmological volumes when conducting the HOD simulations.

We pose this optimization problem because IA often act as a contaminant for weak lensing measurements, and moreover, there are known regions of parameter space that lead to low correlations. Thus, with this experiment, we are able to demonstrate that a differentiable forward model allows one to learn something about the underlying physics of IA. In other words, we are able to recover a known parameter space even without hard-coding the physical laws into the surrogate directly. [3]

The surrogate model introduced in Pandya et al. (2024; 2025b) is a deep neural network with a multilayer perceptron encoder and three convolutional neural network decoder heads for each of the three IA correlation functions. The model is also constructed to predict aleatoric uncertainties per-bin for each correlation, and epistemic uncertainties via the Monte Carlo dropout technique (Hüllermeier & Waegeman, 2021). For a single initialization of the seven Halotools-IA parameters, Figure 11 shows all three IA correlation functions. Our study focuses exclusively on the $\omega(r)$ correlation. This is because $\xi(r)$ is akin to galaxy clustering and therefore does not contain any IA information itself. In addition, the galaxy shape noise that is present in $\eta(r)$ significantly increases the difficulty of extracting relevant signal for gradient-based optimization pipelines, as seen in Figure 11. This leaves us with $\omega(r)$, as it contains IA information but is significantly less obscured by shape noise when compared to $\eta(r)$. This is seen quantitatively in Figure 48, which shows the Jacobian of each correlation as a function of the model inputs. It is seen that the gradient magnitudes of `IAEmu` are in general larger than that of $\eta$. These gradients are also stable from bin-to-bin, and can further be accessed in less than a second. While there may be some disagreement with the exact sensitivity that the Halotools-IA[4] model presented in Van Alfen et al. (2024) would predict, we will see that the gradients information from `IAEmu` further affirms that it is an accurate emulator for Halootols-IA.

To isolate regions of parameter space in $\mu_{\rm cen}$ and $\mu_{\rm sat}$ that minimize the IA correlation $\omega(r)$, we utilize Hamiltonian Monte Carlo (HMC). HMC is a gradient based Monte Carlo technique which leverages the differentiability of NNs and uses gradient information to efficiently traverse the input parameter space towards regions that maximize the posterior probability. For a general review of HMC, see Betancourt (2017). Our HMC experiment mirrors Pandya et al. (2025b). We run our HMC with a No U-Turn Sampler (NUTS) (Hoffman et al., 2014) with 1000 warmup steps and 1000 samples. We fix the HOD parameters according to the fiducial values present in IllustrisTNG300-1 (Nelson et al., 2019) and Table C1 of Van Alfen et al. (2024). This gives us three sets of HOD parameters, which each correspond to a different stellar mass cutoff in galaxies. We impose uniform priors of $[-1, 1]$ on $\mu_{\rm cen}$ and $\mu_{\rm sat}$ and use a multivariate normal distribution centered at zero as our likelihood, with a covariance given by the IAEmu aleatoric uncertainty prediction. In this way, we encourage HMC to arrive at regions of parameter space that minimize the $\omega$ correlation, while also incorporating the inherent shape noise into the likelihood. A general outline of our procedure is given in algorithm 3.

[3] The exact correlation that you would subtract off is the gravitationally lensed shape - intrinsic shape ($GI$) correlation (Lamman et al., 2023); however, the strength of that correlation is closely related to that of $\omega$.

[4] `https://github.com/astropy/halotools`

**Algorithm 3** Surrogate Differentiable Estimation

1: Simulate realizations of a correlation function with astrophysical model parameters $\phi$
2: Divide correlation function outputs into training, validation, and testing sets
3: Train a model with learnable parameters $\theta$ to learn the relationship between astrophysical model parameters and correlation functions from a (not necessarily differentiable) forward model, $\xi_\theta(\phi)$
4: Once trained, perform inference on parameters of choice
5: Optionally optimize via gradient descent the parameters $\phi$ that minimize some loss function $L(\xi_\theta(\phi))$
6: Optionally retrieve via HMC the parameter posteriors $p(\phi|\text{data})$ with priors $p(\phi)$ and a likelihood $p(\text{data}|\phi)$ =0

Figures 12 and 13 confirm that $\mu_{\text{cen}}$ and $\mu_{\text{sat}}$ values of zero minimize $\omega$, with some degeneracy as indicated by the diagonal shape of the posterior. While values of $\mu_{cen} = 0 = \mu_{sat}$ are likely to minimize the correlation function, regions where the signs of $\mu_{cen}$ and $\mu_{sat}$ are opposite can also result in low correlations. Physically, this corresponds to scenarios wherein the negative (i.e. perpendicular) alignments of satellite galaxies can oppose the positive (i.e. parallel) alignment of centrals, resulting a overall small value of $\omega(r)$.

## 8 Summary and Conclusions

In this work, we studied correlation functions along three axes: model uncertainty, differentiability, and surrogates. To better understand the model uncertainty caused by clustering approximations, we used $\rho$ statistics as a case study. We made our assignment of objects to clusters probabilistic, and computed the mean and standard deviation of the resulting correlation estimates. Our analysis compared the uncertainty estimate to traditional bootstrapping methods. Our next experiment adapted our main algorithm to be automatically differentiable and applied this to the setting of gravitational simulations. Three different approaches of doing this were proposed, and of those, approach $c$ proved to be the most stable. Still, the calculation of the gradient was slow and approximate, motivating the use of surrogate methods. To that end, we ended by exploring surrogate solutions. We extended the work of Pandya et al. (2024; 2025b), showing how we can exploit the gradients of our surrogate model to optimize over astrophysical model parameters. Specifically, we studied galaxy intrinsic alignment, and found which astrophysical model parameters were most responsible for high / low correlations (Figures 12-13).

On uncertainty, it was found that the process of clustering objects before computing correlations can cause measurable uncertainties in limited data settings, such as with the COSMOS-Web PSF modeling efforts. That is, the model uncertainty was just as severe as the data uncertanties found via bootstrapping. In these circumstances, we conclude that the naïve approach, which considers all distance pairs without clustering to be most appropriate. Moreover, we find that our method of probabilistic clustering allows us to address one of the fundamental concerns in uncertainty quantification, that being the distinction between epistemic and aleatoric uncertainties. Again, this is quantified in Figures 8-47, where the error bars for epistemic and aleatoric uncertainties are shown for each of the five $\rho$ statistics.

On differentiability, we showed that the simplest differentiable algorithm has features that makes the computation of the gradients slow and unstable. These findings were demonstrated through attempts to differentiate a correlation function with respect to a single parameter $G$ that determined the results of a gravitational simulation. Three approaches, a-c, were outlined: Approach $a$ relied on the Gumbel-Max trick, approach $b$ used the weight matrix to compute new quantities via weighted averages, and approach $c$ used an approximate method to calculate the gradient. Of these, approach $c$ proved to be the most effective at circumnavigating these deficiencies. The features that caused slowness and instability were the normalization and distance function calls and sparse Jacobians that propagated through the chain rule. The normalization and distance function calls posed challenges for fast automatic differentiation as they cannot be expressed simply with matrix multiplication; the normalization function calls in particular are notorious bottlenecks for automatic

differentiation pipelines. We also saw that many of our function calls yielded sparse Jacobians. Since the chain rule amounts to multiplying successive Jacobians, this caused numerical instabilities in computing the final derivative. It is worth noting that multiplying one matrix by a sparse matrix does not necessarily imply that the resulting matrix is sparse. However, through testing the function calls individually, we found that the numerical instabilities happened only once the gradients were multiplied together, indicating that matrix sparsity patterns in any one Jacobians is the culprit for numerical instability. This also explained why approach $c$ was most successful, as approach $c$ avoided the function call that related weights to the input data when taking the gradient. Approach $c$ was successfully able to capture the gradient of a correlation function with respect to an input cosmology. While there remains future work in optimizing approach $c$ for GPU support and overall making it faster, it proved to be the most effective way to write down correlation function estimators that are automatically differentiable. In the main, the differentiability experiment suggested that a more judicious way to construct differentiable correlation functions may be to use a surrogate model.

On surrogates, we found that surrogate models are a simple and effective way to optimize astrophysical model parameters that determine a correlation function. Using galaxy IA as an example, a correlation function was minimized across 20 radial bins. Using HMC, we were able to recover a known posterior over the IA parameters that lead to low correlation function outputs. A key caveat is that the true correlation functions that act as a contaminant are the $GI$ and $gI$ correlations (as defined in Lamman et al., 2023), but the $\omega(r)$ correlation is closely related in how it represents the presence of IA.

Prior to our work, properties such as model uncertainty and differentiability have been understudied in the context of correlation function estimators. This work is intended as a first step toward addressing these points. Our study is intentionally generic — survey systematics can vary tremendously and the correlations being studied will also vary by science case. Rather than focusing on a specific scientific inquiry, we propose general algorithms and analyze their utility across different domains. Our GitHub artifact outlines and implements these algorithms, serving as a blueprint for future large-scale surveys in each of the contexts laid out in this work — model uncertainty, differentiability, and surrogates.

## Part II

# On Differentiable Correlation Functions

Traditionally in the cosmology literature, 2-point correlations of galaxy properties are studied in terms of their dependence on spatial separations. The values of these 2-point correlation functions also carry a dependence on underlying cosmology. Motivated by the desire to study the dependence of correlation functions on cosmological parameters, we introduce an algorithm for estimating correlations that is automatically differentiable by design. Automatic differentiation allows us to quantify the relationship between a correlation and a cosmology through studying the derivatives of the correlation function with respect to each of the input parameters in the model. The widely adopted KD-Tree and k means approaches usually use the argmin and median functions to cluster objects. These functions are either non-differentiable or result in a derivative of 0, losing the information we wish to study. In contrast, our approach uses fuzzy c means to cluster objects into representative groups, keeping computational expediency without losing differentiability. We test our algorithm by time evolving galaxies randomly distributed in a gravitational field until some stopping time Tmax and then running our algorithm on the resulting catalog. We then take the derivative of our correlation function with respect to various values of the gravitational constant G. This proof of concept shows the efficacy of our algorithm for studying the dependencies of correlation functions on cosmology.

# Part III

# On Uncertainty Calibration for Equivariant Functions


## Abstract

Data-sparse settings such as robotic manipulation, galaxy morphology classification, and chemical physics are some of the hardest domains for deep learning. For these problems, equivariant networks can help improve modeling across undersampled parts of the input space, and uncertainty estimation can guard against overconfidence. However, until now, the relationship between equivariance, model calibration, and model confidence has yet to be studied. In this work, we present the first theory unifying equivariance and uncertainty estimation. By proving upper bounds on uncertainty calibration errors under various assumptions of equivariance, we study the generalization limits of equivariant models and illustrate how symmetry mismatch can result in miscalibration. We study the impact of these bounds on a variety of real and simulated experiments, and we comment on trends with correct, incorrect, and extrinsic equivariance, group order, and aleatoric and epistemic uncertainties.


## 9 Introduction

A surprising result of (Wang et al., 2023a) is that equivariant models can still be effective even in cases of mismatch between the model and the data symmetry. This finding motivated the work of Wang et al. (2024), which explored how equivariance can affect model *accurancy*, both positively and negatively. However, it is not yet understood how equivariance impacts model *calibration*, loosely defined as the disagreement between a model's accuracy and predicted *confidence*. Understanding both model calibration and confidence is particularly useful in the data-sparse settings where equivariant neural networks tend to thrive, such as pick-and-place robotics tasks (Kalashnikov et al., 2018; Wang et al., 2022b;a; Fu et al., 2023; Huang et al., 2023; 2024b;a), galaxy morphology classification (Pandya et al., 2023; 2025a), and molecular physics (Zou et al., 2023b; Ramakrishnan et al., 2014). While equivariance has proved invaluable in these scenarios, equivariance is not a golden bullet: equivariance comes with well-defined tradeoffs, including fairly limited benefits at scale (Wang et al., 2023b; Klee et al., 2023; Gruver et al., 2023; Brehmer et al., 2024; Abramson et al., 2024), provable degradation on model performance in cases of symmetry mismatch (Wang et al., 2024), more complex architectures, and the need for more compute. To help quantify these tradeoffs, we seek to study how the inductive bias of equivariance affects different metrics such as calibration error. This in turn allows us to address several unanswered questions on the subject of calibration and confidence of equivariant models. Namely, when does equivariance help a model predict its own confidence? How do notions of correct, incorrect, and extrinsic (Wang et al., 2024) equivariance (the equivariance taxonomy hereafter) affect model calibration? What is the general relationship between equivariance and uncertainty estimation? For the purposes of claiming a scientific discovery or avoiding disaster when handling expensive and brittle equipment, model calibration is very often more important than model performance, and the effect of equivariance on such calibration error remains unclear.

The purpose of this work is to address both the lack of a unified theory relating equivariance to uncertainty estimation, and the absence of experiments exploring this relationship in practice. To accomplish this, we extend the error bounds given by Wang et al. (2024) to a broader class of calibration losses. In this way, we can quantify the effect of equivariance not just on accuracy, but also on calibration. The only works that really explore the direct relationship between equivariance and uncertainty are Sun et al. (2023) and Cherif et al. (2024). Accordingly, we supplement our bounds with experiments on a wide variety of real and simulated datasets that indicate how these results manifest. Importantly, we show that our bounds can be realized in a true experimental setting. We summarize our contributions as follows:

1. We provide general bounds on how equivariance affects calibration error for classification tasks. We further examine the limiting cases of over and under confidence in equivariant models (§12.1).

2. We generalize the expected normalized calibration error (Equation 68) beyond scalar values for mean and variance predictions and derive its theoretical upper bound on equivariant models (§12.2-12.3). We further study our proposed metric in the limiting case of minimized regression error using the bounds from Wang et al. (2024). Additionally, we coin another metric, the aleatoric bleed, in order to study miscalibration in terms of aleatoric and epistemic uncertainty (§12.4).

3. In the effort to better understand the impact of our bounds, we explore the effect of equivariance on model calibration, confidence, and aleatoric bleed using a wide variety of real and simulated datasets (§13).

4. We examine trends in group order, performance on data within different regimes of the equivariance taxonomy, and the ability to disaggregate aleatoric and epistemic uncertainties (§13.1 - 13.4).

5. We compare the effect of binning approximations on our generalized expected normalized calibration error with the results of Pernot (2023) on QM9s data (Ramakrishnan et al., 2014) (§13.5).

6. Following the discussion of §13.5, we show that regardless of the number of bins used to approximate the true calibration error, the upper bound on the expected normalized calibration error can be realized on real datasets. Specifically, we do this on rotation augmented versions of QM9s (§13.6).

We publicly release all code and datasets used in this work at .

## 10 Related Work

**Equivariant Learning.** Our work is closest in flavor to Petrache & Trivedi (2023) and Wang et al. (2024), which both establish bounds on function generalization under various assumptions of symmetry mismatch. We adopt many of the same formalisms as Wang et al. (2024), which was motivated by Wang et al. (2023a) and takes theoretical inspirations from Finzi et al. (2020). Our work makes use of the assumption that equivariant functions are universal approximators, which was shown to be true for $G-$equivariant functions in Maron et al. (2019); Yarotsky (2022). Our work also uses the strategy of decomposing the input and output spaces in order to prove some bounds, which we accomplish through taking the quotient by a group. This is a strategy similarly employed in Sannai et al. (2021); Lawrence (2022); Petrache & Trivedi (2023); Wang et al. (2024), with the formulation of Petrache & Trivedi (2023) sharing the most commonalities with our work. Theory on equivariant learning is partly addressed for probabilistic symmetries in Bloem-Reddy et al. (2020), but no bounds on calibration error are presented. Reasoning about distributions in terms of invariants also has a rich history in arguing why one prior is better than another for Bayesian analysis – see Jaynes (1968). We build on this by studying how equivariance can affect the reliability of Bayesian methods. Specifically, we look at the ability to dissagregate different types of uncertainty using evidential regression, including on multivariate distributions, generalizing some of the work of Van der Linden et al. (2025) who study equivariant model selection when trained to predict a univariate distribution.

**Aleatoric and Epistemic Uncertainties.** A longstanding goal in the computational sciences is to disaggregate model (epistemic) uncertainties from (aleatoric) uncertainties inherent to the data (Ulmer et al., 2021; Hüllermeier & Waegeman, 2021; Osband et al., 2023; Fuchsgruber et al., 2024). A key distinction is that epistemic uncertainties can be reconciled with more data, whereas aleatoric uncertainties can not. Whereas previous works have no explored the relationship between symmetry and uncertainty, this work explores how the epistemic uncertainty, the uncertainty often quantified by calibration errors, can be confused with aleatoric uncertainty due to symmetry mismatch. We note that in the sciences, these are often coined as "systematic" and "statistical" uncertainties, e.g. Freedman et al. (2024).

**Learning Parameterized Distributions.** For many learning tasks, it is natural to design a neural network that approximates a probability distribution rather than a single point estimation. There are many ways of doing this, such as with Bayesian Neural Networks (Kononenko, 1989), Epistemic Neural Networks (Osband et al., 2023), Normalizing Flows (Kobyzev et al., 2020; Papamakarios et al., 2021), or simply employing the softmax (Goodfellow et al., 2016). These approaches are often computationally expensive to compute in practice. Thus, we often employ parameterization techniques to constrain neural networks to output simplified probability distributions and train them using a negative log likelihood loss derived from said distribution. The parameters that define the output distributions are easy to interpret and are often easier to learn than aforementioned alternative approaches. Mean variance estimators (MVE) are the simplest type of neural network that predicts a parameterized distribution. Instead of predicting a single output, they predict a mean $\mu$ and a variance $\sigma^2$ (Nix & Weigend, 1994; Seitzer et al., 2022). There is also work extending MVEs to learn covariances for multivariate distributions (Tomczak et al., 2020) and linear combinations of Gaussians (Diakonikolas et al., 2020). Amini et al. (2020) extend MVEs by imposing a prior on $\mu$ and $\sigma^2$, which in turn provides enough parameters to disentangle aleatoric and epistemic uncertainties. Hüttel et al. (2023) further extended Amini et al. (2020) so that one can also learn different quantiles of the distribution as a way of learning aleatoric uncertainty.

**Calibration Error.** It is common in classification tasks to choose the target label with the highest probability. However, even if a model gives a label $y$ the highest probability $p$, that does not mean the model will necessarily be correct with probability $p$. This mismatch is often quantified with the expected calibration error (ECE) (Guo et al., 2017). Miscalibration can analogously be measured for regression tasks (Pernot, 2023; Levi et al., 2022b). The idea is to compare true labels $y$ with a predicted mean $\mu$ and variance $\sigma^2$ of a MVE. One should expect the squared errors $(y-\mu)^2$ to average out to the variance $\sigma^2$. This idea was made precise by Levi et al. (2022a), who proposed the expected normalized calibration error (ENCE) to quantify that exact discrepancy. A key limitation of their work is that it is formulated in terms of binning approximations rather than in terms of a continuous probability density. Another key limitation of their work is that they assume mean and variance are scalar values. The work of Sun et al. (2023) suggests that equivariance can improve model calibration, but a theoretical justification for this is not present in the literature. Beyond ECE and ENCE, we also explore calibration in terms of coverage, for which we refer the reader to Gneiting & Raftery (2007); Sun et al. (2023).

## 11 Background

### 11.1 Equivariance

Here, we give precise definitions of equivariance and invariance, following §A.2 in Brandstetter et al. (2021). For a general review of abstract algebra and representation theory, we direct the reader towards Artin (1998); Esteves (2020); Hall (2013).

Equivariance is a property of an operator $\phi : \mathcal{X} \to \mathcal{Y}$ that maps between input and output vector spaces $\mathcal{X}$ and $\mathcal{Y}$. Given a group $G$ and its representations $\rho^{\mathcal{X}}$ and $\rho^{\mathcal{Y}}$ which transform vectors in $\mathcal{X}$ and $\mathcal{Y}$ respectively, an *operator $\phi : \mathcal{X} \to \mathcal{Y}$ is said to be equivariant if it satisfies the following constraint*

$$\rho^{\mathcal{Y}}(g)[\phi(x)] = \phi(\rho^{\mathcal{X}}(g)[x]) \ , \ \text{for all } g \in G, x \in \mathcal{X} \ . \tag{24}$$

Where we use $gx$ is it implicit that we are considering the representation $\rho(g)$, an $n \times n$ matrix, for $g \in \mathbf{G}$. This means that $gx$, the result of the group action acting on $x$, is a vector.

Invariance is a special case of equivariance in which $\rho^{\mathcal{Y}} = \mathcal{I}^{\mathcal{Y}}$ for all $g \in G$. I.e., *an operator $\phi : \mathcal{X} \to \mathcal{Y}$ is said to be invariant if it satisfies the following constraint*

$$\phi(x) = \phi(\rho^{\mathcal{X}}(g)[x]) \ , \ \text{for all } g \in G, x \in \mathcal{X} \ . \tag{25}$$

Thus, with an invariant operator, the output of $\phi$ is unaffected by transformations applied to the input.

### 11.2 Notation and Main Results from Wang et al. (2024)

Given our expressed goal of generalizing the bounds from Wang et al. (2024) and our heavy reliance on their formalism, we review the relevant notation and main results here. We start by restating the main problem statement.

**Problem Statement.** Consider a function $f : X \to Y$. Let $p : X \to \mathbb{R}$ be the probability density function of the domain $X$. We assume that there is no distribution shift during testing, i.e., $p$ is always the underlying distribution during training and testing. The goal for a model class $\{h : X \to Y\}$ is to fit the function $f$ by minimizing an error function $\text{err}(h)$. We assume the model class $\{h\}$ is arbitrarily expressive except that it is constrained to be equivariant with respect to a group $G$. Let $\mathbb{I}$ be an indicator function that equals to 1 if the condition is satisfied and 0 otherwise. In classification, $\text{err}(h)$ is the classification error rate; for regression tasks, the error function is a L2 norm function,

$$\text{err}_{\text{cls}} = \mathbb{E}_{x\sim q}\left[\mathbb{I}(f(x) \neq h(x))\right] \tag{26}$$

$$\text{err}_{\text{reg}} = \mathbb{E}_{x\sim q}\left[\left\|h(x) - f(x)\right\|_2^2\right]. \tag{27}$$

#### 11.2.1 A Taxonomy of Equivariance

**Definition 1** (**Correct Equivariance, Definition 3.1 in Wang et al. (2024)**). For all $x \in X$, $g \in G$ where $p(x) > 0$, if $p(gx) > 0$ and $f(gx) = gf(x)$, $h$ has *correct equivariance* with respect to $f$.

**Definition 2** (**Incorrect Equivariance, Definition 3.2 in Wang et al. (2024)**). For all $x \in X$, $g \in G$ where $p(x) > 0$, if $p(gx) > 0$ and $f(gx) \neq gf(x)$, $h$ has *incorrect equivariance* with respect to $f$.

**Definition 3** (**Extrinsic Equivariance, Definition 3.3 in Wang et al. (2024)**). For all $x \in X$, $g \in G$ where $p(x) > 0$, if $p(gx) = 0$, $h$ has *extrinsic equivariance* with respect to $f$.

**Definition 4** (**Pointwise Correct Equivariance, Definition 3.5 in Wang et al. (2024)**). For $g \in G$ and $x \in X$ where $p(x) \neq 0$, if $p(gx) \neq 0$ and $f(gx) = gf(x)$, $h$ has correct equivariance with respect to $f$ at $x$ under transformation $g$.

**Definition 5** (**Pointwise Incorrect Equivariance, Definition 3.6 in Wang et al. (2024)**). For $g \in G$ and $x \in X$ where $p(x) \neq 0$, if $p(gx) \neq 0$ and $f(gx) \neq gf(x)$, $h$ has incorrect equivariance with respect to $f$ at $x$ under transformation $g$.

**Definition 6** (**Pointwise Extrinsic Equivariance, Definition 3.7 in Wang et al. (2024)**). For $g \in G$ and $x \in X$ where $p(x) \neq 0$, if $p(gx) = 0$, $h$ has extrinsic equivariance with respect to $f$ at $x$ under transformation $g$.

#### 11.2.2 Regression Bounds

**Theorem 1** (**Theorem 4.8 in Wang et al. (2024)**). We assume $h$ is $G$ invariant so that $h(gx) = h(x)$ for all $g \in G$. Assume $Y = \mathbb{R}^n$. Denote by $p(Gx) = \int_{z\in Gx} p(z)dz$ the probability of the orbit $Gx$. Denote by $q(z) = \frac{p(z)}{p(Gx)}$ the normalized probability density of the orbit $Gx$ such that $\int_{Gx} q(z)dz = 1$. Let $\mathbb{E}_{Gx}[f]$ be the mean of the function $f$ on the orbit $Gx$ defined, and let $\mathbb{V}_{Gx}[f]$ be the variance of $f$ on the orbit $Gx$,

$$\mathbb{E}_{Gx}[f] = \int_{Gx} q(z)f(z)dz = \frac{\int_{Gx} p(z)f(z)dz}{\int_{Gx} p(z)dz} \tag{28}$$

$$\mathbb{V}_{Gx}[f] = \int_{Gx} q(x)\|\mathbb{E}_{Gx}[f] - f(z)\|_2^2. \tag{29}$$

We have $\text{err}(h) \geq \int_F p(Gx)\mathbb{V}_{Gx}[f]$.

**Theorem 2** (**Theorem 4.9 in Wang et al. (2024)**). We now only assume equivariance on $h$, that is, $h(\rho_X(g)x) = \rho_Y(g)h(x)$ where $g \in G$, $\rho_X$ and $\rho_Y$ are group representations associated with $X$ and $Y$. We will denote $\rho_X(g)x$ and $\rho_Y(g)y$ by $gx$ and $gy$, leaving the representation implicit. Assume $Y = \mathbb{R}^n$. Let Id be the identity. Define a matrix $Q_{Gx} \in \mathbb{R}^{n\times n}$ and $q(gx) \in \mathbb{R}^{n\times n}$ so that $\int_G q(gx)dg = \text{Id}$ by

$$Q_{Gx} = \int_G p(gx)\rho_Y(g)^T\rho_Y(g)\alpha(x,g)dg \tag{30}$$

$$q(gx) = Q_{Gx}^{-1}p(qx)\rho_Y(g)^T\rho_Y(g)\alpha(x,g). \tag{31}$$

If $f$ is equivariant, $g^{-1}f(gx)$ is a constant for all $g \in G$. Define $\mathcal{E}_G[f,x]$

$$\mathcal{E}_G[f,x] = \int_G q(gx)g^{-1}f(gx)dg. \tag{32}$$

The error of $h$ has lower bound $\text{err}(h) \geq \int_F \int_G p(gx)\|f(gx) - g\mathcal{E}_G[f,x]\|_2^2\alpha(x,g)dgdx$.

### 11.3 Evidential Regression

The appeal of (deep) evidential regression for our work is that it allows us to precisely define epistemic and aleatoric uncertainties. Here, we review the relevant theory and notation from Amini et al. (2020).

Given $(y_1, \ldots, y_n) \sim \mathcal{N}(\mu, \sigma^2)$, we may impose priors

$$\mu \sim \mathcal{N}(\gamma, \sigma^2\nu^{-1}) \tag{33}$$

$$\sigma^2 \sim \Gamma^{-1}(\alpha, \beta) \tag{34}$$

where $\Gamma(\cdot)$ is the gamma function. Let $m = (\gamma, \nu, \alpha, \beta)$, and $\gamma \in \mathbb{R}$, $\nu > 0$, $\alpha > 1$, $\beta > 0$. One can then show that $p(y_i|m) = St(y_i; \gamma, \frac{\beta(1+\nu)}{\alpha\nu}, 2\alpha)$, where the St distribution given by

$$St(t;\mu,\sigma,\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu}\sigma\Gamma\left(\frac{\nu}{2}\right)}\left(1 + \frac{1}{\nu}\left(\frac{t-\mu}{\sigma}\right)^2\right)^{-(\nu+1)/2}. \tag{35}$$

Parameterizing the St distribution as a four parameter family is useful because it allows us to define our prediction, aleatoric uncertainty, and epistemic uncertainty in a rigorous way:

$$\mathbb{E}[\mu] = \gamma \quad \text{(Prediction)} \tag{36}$$

$$\mathbb{E}[\sigma^2] = \frac{\beta}{\alpha - 1} \quad \text{(Aleatoric Uncertainty)} \tag{37}$$

$$\text{Var}[\mu] = \frac{\beta}{\nu(\alpha-1)} \quad \text{(Epistemic Uncertainty)}. \tag{38}$$

With some slight abuse of notation, we may abbreviate these as $\sigma^2_{\text{aleatoric}} := \mathbb{E}[\sigma^2]$ and $\sigma^2_{\text{epistemic}} := \text{Var}[\mu]$. Some works take these uncertainties to be additive. That is, the model will predict $\gamma \pm \sigma^2_{\text{aleatoric}} \pm \sigma^2_{\text{epistemic}}$. See Freedman et al. (2024) and Berman & McCleary (2025b); Berman et al. (2025) for examples where this is done in the sciences. We do not assume these uncertainties to be additive in this work.

## 12 Theoretical Results

In this section, we present our main theoretical results, including our bounds on uncertainty calibration error for classification and regression cases on invariant and equivariant functions. Throughout the problem

statements, when we introduce a set, vector space, or group, we will give it a boldface, such as $\mathbf{E}$; however, this notation will be dropped in the theorems and proofs. In the exposition that follows, we stagger problem statements with their relevant theorems.

### 12.1 Invariant Classification Upper Bounds

We lead with classification. We show that the expected calibration error as it is typically defined is a normalized metric. However, when we consider invariance, we can put bounds on when a model is over and under confident, and we can illuminate a circumstance where the upper bound on expected calibration error is tighter for invariant models.

**Classification Problem Setup.** Consider a function $f : \mathbf{X} \to \mathbf{Y}$ where $\mathbf{Y}$ is a finite set of labels. Let $q : \mathbf{X} \to \mathbb{R}$ be the probability density of the domain $\mathbf{X}$. Now, we define a model class as the set $\{h : \mathbf{X} \to \mathbf{Y} \times \hat{\mathbf{P}}\}$ where $\hat{\mathbf{P}} = [0, 1]$ with elements $p$ representing the normalized confidence. We will denote the two outputs by $h_Y$ and $h_P$. The goal for our model class is to fit the function $f$ *and* to properly predict its own confidence by minimizing the expected calibration error (Equation 39, and Equation 1 in Guo et al. (2017)). Following Wang et al. (2024), we assume that the model class $\{h\}$ is arbitrarily expressive except that it is constrained to be equivariant with respect to a group $G$. For this classification setting, we specifically choose $h$ to be $G-$invariant. Let $\mathbb{I}$ be an indicator function that equals 1 if the condition is satisfied and 0 otherwise. Let $r(p)$ be the probability density such that $\mathbb{P}(p_1 \leq h_P(x) \leq p_2) = \int_{p_1}^{p_2} r(p)dp$.

$$\text{ECE}(h) = \mathbb{E}_{h_P}\left[\left|\, \mathbb{P}\left(f = h_Y | h_P = p\right) - p \,\right|\right]. \tag{39}$$

Additionally, we will use the concepts of iterated integration and fundamental domains, which we review in Appendix A. We will also use the equivariance taxonomy, which we reviewed in §11.2.1.

We now show that ECE is a normalized metric and discuss limiting caes of over and under confidence, which we will later expand on when we introduce an assumption of invariance on $h$.

**Theorem 3.** Equation 39 is a normalized metric, it is bounded below by 0 and is bounded above by 1. When a model is consistently overconfident ($\mathbb{P}\left(f = h_Y | h_P = p\right) < p$ for all $p$), then the upper bound on ECE is given by $\mathbb{E}_{x\sim q}[h_P]$. Similarly, when a model is consistently underconfident ($\mathbb{P}\left(f = h_Y | h_P = p\right) > p$ for all $p$), then the upper bound on ECE is given by $1 - \mathbb{E}_{x\sim q}[h_P]$.

*Proof.* We begin by rewriting Equation 39 as

$$\mathbb{E}_{h_P}\left[\left|\, \mathbb{P}\left(f = h_Y \;\middle|\; h_P = p\right) - p \,\right|\right] = \int_{p=0}^{p=1} r(p) \left|\, \mathbb{P}\left(f(x) = h_Y(x) \;\middle|\; h_P(x) = p\right) - p \,\right| dp. \tag{40}$$

For constants $A$ and $B$, the triangle inequality gives us $0 \leq |A - B| \leq |A| + |B|$. If both $A$ and $B$ are nonnegative then $|A - B| \leq \max\{A, B\}$ with equality if and only if one of $A$ or $B$ is zero. Since $\mathbb{P}\left(f = h_Y \;\middle|\; h_P = p\right)$ and $p$ are non-negative, we can further restrict the inequality to be

$$0 \leq \int_{p=0}^{p=1} r(p) \left|\, \mathbb{P}\left(f(x) = h_Y(x) \;\middle|\; h_P(x) = p\right) - p \,\right| dp \tag{41}$$

$$\leq \int_{p=0}^{p=1} r(p) \max\{\mathbb{P}\left(f(x) = h_Y(x) | h_P(x) = p\right), p\} dp. \tag{42}$$

Recall that we said $\{h\}$ is arbitarily expressive except for that it is invariant with respect to a group $G$. Thus, we can always choose $h_Y$ and $h_P$ such that

$$\left|\, \mathbb{P}\left(f(x) = h_Y(x) \;\middle|\; h_P(x) = p\right) - p \,\right| = 0. \tag{43}$$

This gives us our lower bound. We will now proceed to show the upper bound. We will denote by $c$ the original integral,

$$c := \int_{p=0}^{p=1} r(p) \left| \mathbb{P}\left(f(x) = h_Y(x) \;\middle|\; h_P(x) = p\right) - p \right| dp. \tag{44}$$

We may abbreivate $\max\{\mathbb{P}\left(f(x) = h_Y(x) | h_P(x) = p\right), p\}$ as $\max(p)$. Now consider the following three cases:

**Case 1: The model h is always overconfident.** In this case, $\max(p) = p$ for all $p$.

$$\begin{aligned} c &\leq \int_{p=0}^{p=1} r(p) \max(p) dp && (45)\\ &= \int_{p=0}^{p=1} r(p) p dp && (46)\\ &= \mathbb{E}[h_P] && (47) \end{aligned}$$

Notice that we get this result exactly if we take $\mathbb{P}\left(f(x) = h_Y(x) \;\middle|\; h_P(x) = p\right) = 0$ in Equation 39.

**Case 2: The model h is always underconfident.** In this case, $\max(p) = \mathbb{P}\left(f(x) = h_Y(x) | h_P(x) = p\right)$ for all $p$. It was revealing that the upper bound in the prior case came about from $\mathbb{P}\left(f(x) = h_Y(x)\right) = 0$. Similarly, we will find that the upper bound in this case comes about from $\mathbb{P}\left(f(x) = h_Y(x)\right) = 1$.

Since we assumed that there are finitely many labels in the codomain $Y$, we can assume that $\mathbb{P}\left(h_Y(x) | h_P(x) = p\right)$ is a discrete probability distribution for each value of $p$. Therefore, each outcome in the distribution has a probability less than one. Accordingly, the outcome $\mathbb{P}\left(f(x) = h_Y(x) | h_P(x) = p\right)$ also has a probability less than one. So, $|\mathbb{P}\left(f(x) = h_Y(x) | h_P(x) = p\right) - p|$ is maximized when $\mathbb{P}\left(f(x) = h_Y(x) | h_P(x) = p\right)$ attains its upper bound of one for each $p$ on $[0, 1]$.

We compute

$$\begin{aligned} c &= \int_{p=0}^{p=1} r(p) \left| 1 - p \right| dp && (48)\\ &= \int_{p=0}^{p=1} r(p)(1 - p) dp && (49)\\ &= 1 - \mathbb{E}[h_P]. && (50) \end{aligned}$$

Notice the parallel with Equation 47. This symmetry can be attributed to the nature of the absolute value and the fact that the limiting values occur when $\text{err}_{\text{cls}}(h)$ is zero or one, see Figure 14.

In Appendix J, we show how the bound can be related to traditional classification error.

**Case 3: The model h is sometimes overconfident and sometimes underconfident.** In this case, $\max(p)$ will vary with $p$. Let $p_1$ be the domain $[0, 1]$ restricted to regions where the model $h$ is overconfident, and similarly let $p_2$ be the domain $[0, 1]$ restricted to regions where the model $h$ is underconfident.

$$c \leq \int_{p=0}^{p=1} r(p)\max(p)dp \tag{51}$$

$$= \int_{\substack{p\in p_1 \\ \leq \mathbb{E}[h_P]}} r(p)pdp + \int_{p\in p_2} r(p)\mathbb{P}\left(f(x) = h_Y(x) \;\middle|\; h_P(x) = p\right)_{\leq 1-\mathbb{E}[h_P]} dp \tag{52}$$

$$\leq \mathbb{E}[h_P] + (1 - \mathbb{E}[h_P]) \tag{53}$$

$$= 1 \tag{54}$$

We conclude that $c$, and therefore $\text{ECE}(h)$, is upper bounded by 1. Under the assumptions of over and under confidence, the upper bounds become $\mathbb{E}[h_P]$ and $1 - \mathbb{E}[h_P]$ respectively. □

**Remark 1.** Note that our proof for case 2 depends upon the assumption that there are finitely many labels in the codomain $Y$.

Now that we have an understanding of the bounds on ECE, we seek to understand how the assumption of model invariance can alter this bound. For a model class $h$ that is correctly invariant, we can obtain perfect accuracy and perfect calibration. What is perhaps more interesting is the case where our model is incorrectly invariant.

**Corollary 1.** Assume that $h$ is incorrectly invariant under a finite group $G$ on our input domain $X$. Let $\mathcal{F}_p$ be a fiber given by $h_P^{-1}(\{p\})$ with elements $x_p$, let $b_n(p)$ be the probability $\mathbb{P}(x_p \in \{f(x_p) = y_n\})$, where $y_n$ is a label in $Y$. Let $k$ be given by $\min \sum_{n=1}^{N} b_n(p)\frac{n-1}{|G|}$ on $p_2$ where $p_2$ is as defined in Theorem 3. With the incorrect invariance constraint on $h$, ECE becomes upper bounded by $\mathbb{E}[h_p] + 1 - k$.

*Proof.* We start by the revisiting the bound on $c$ from Theorem 3:

$$c \leq \int_{p=0}^{p=1} r(p)\max(p)dp \tag{55}$$

$$= \int_{\substack{p_1 \\ \leq \mathbb{E}[h_P]}} r(p)pdp + \int_{p_2} r(p)\mathbb{P}\left(f(x) = h_Y(x) \;\middle|\; h_P(x) = p\right) dp. \tag{56}$$

From proposition A1 in Wang et al. (2023a), the accuracy on any fiber of $p \in p_2$ is upper bounded by $1 - \sum_{n=1}^{N} b_n(p)\frac{n-1}{|G|}$. See that

$$\int_{p\in p_2} r(p)\mathbb{P}\left(f(x) = h_Y(x) \;\middle|\; h_P(x) = p\right) dp \leq \int_{p\in p_2} r(p)\left(1 - \sum_{n=1}^{N} b_n(p)\frac{n-1}{|G|}\right) dp \tag{57}$$

$$\leq \int_{p\in p_2} r(p)(1-k)dp \tag{58}$$

$$= (1-k)\int_{p\in p_2} r(p)dp \tag{59}$$

$$\leq (1-k)\,. \tag{60}$$

Therefore, $c \leq E[h_P] + 1 - k$. This completes the proof. □

**Remark 2.** If $k > \mathbb{E}[h_P]$, then this bound becomes tighter than the case of an unconstrained model as in Theorem 3.

The insight here is that the assumption of invariance allows us to upper bound the accuracy on certain level sets, which in turn affects the upper bound on ECE. More generally, we would like to understand how invariance can affect model miscalibration without any prior knowledge on whether the equivariance is correct, incorrect, or extrinsic. We now provide a statement that elaborates on the effect of equivariance on over and under confidences.

**Corollary 2.** Let us restrict Equation 39 to be invariant under $G$, with no restrictions on the type of equivariance. Recall that the region of overconfidence has a calibration error bounded from above by $\mathbb{E}[h_P]$ and the region of underconfidence has a calibration error bounded from above by $1-\mathbb{E}[h_P]$. Denote by $k(Gx)$ the expression $k(Gx) = \max_{p' \in \hat{P}} \int_{z \in Gx} q(z)p'dz$. Then $\mathbb{E}[h_P]$ attains an upper bound of $\int_{x \in F} k(Gx)dz$. Accordingly, the lower bound on $1-\mathbb{E}[h_P]$ is given by $1-\int_{x \in F} k(Gx)dz$. The calibration error on the regions of over and under confidence satisfy these respective bounds.

*Proof.* Let $p'$ be the probability output of $h_P$ on a given orbit $Gx$. This proof will now use iterated iteration over a group, and we remind the reader to see Appendix A for a review of the iterated integration formalism. We can express the expected value of $h_P$ as

$$\mathbb{E}[h_p] = \int_X q(x)h_P(x)dx. \tag{61}$$

Then, we can use iterated integration to show

$$\int_X q(x)h_P(x)dx = \int_{x \in F} \int_{z \in Gx} q(z)h_P(z)dzdx. \tag{62}$$

We can bound the inner integral by looking at the label $h_Y(x) = y_n$ that corresponds to the highest probability $h_P(z) = p'$ on each orbit. In other words, we take the $\max p'$ and corresponding label $\operatorname{argmax}$ $y_n$. This gives us

$$\int_{z \in Gx} q(z)h_P(z)dz \leq \max_{p' \in \hat{P}} \int_{z \in Gx} q(z)p'dz. \tag{63}$$

Assuming $|Y| = n$ labels and softmax normalization given by

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\Sigma_{j=1}^n e^{z_j}}, \tag{64}$$

$p'$ is given by $\max \sigma(\mathbf{z})$. We prove the corollary by integrating over $F$. □

### 12.2 Invariant Regression Upper Bounds

We now study the case of invariant regression. In the process, we define a notion of calibration error that works for vector-valued functions. We find that we are able to bound the calibration error by an expression that is analogous to the maximum $\chi^2$ one can obtain averaged over all of the fibers of $s \in \mathcal{S}$.

**Invariant Regression Problem Setup.** Consider a function $f : \mathbf{X} \to \mathbf{Y}$ where $\mathbf{Y} = \mathbb{R}^n$. Now, we define a model class as the set $\{h : \mathbf{X} \to \boldsymbol{\mathcal{M}} \times \boldsymbol{\mathcal{S}}\}$ where $m \in \boldsymbol{\mathcal{M}} = \mathbb{R}^n$ represents the space of all mean-vectors and $s \in \boldsymbol{\mathcal{S}} = \mathbb{R}^n_+$ represents the space of all variance-vectors. We will denote the two outputs by $h_\mu$ and $h_{\sigma^2}$. Let $p : \mathbf{X} \to \mathbb{R}$ be the probability density over the domain $\mathbf{X}$. Denote the subdomain of $\mathbf{X}$ given by the constraint $h_{\sigma^2}(\vec{x}) = \vec{\sigma^2}$ as $\mathbf{X_s} = \{x \in \mathbf{X} | h_{\sigma^2}(x) = s\}$. We will denote the fundamental domain of $\mathbf{G}$ in $\mathbf{X_s}$ as $\mathbf{F_s}$. We assume that each fundamental domain $\mathbf{F_s}$ satisfies the condition that the union of all pairwise intersections $\cup_{g_1 \neq g_2}(\mathbf{g_1 F_s} \cap \mathbf{g_2 F_s})$ have measure 0. We also assume that $\mathbf{F_s}$ and $\mathbf{G x_s}$ are differentiable manifolds for any $\mathbf{X_s}$ and for all $x_s \in \mathbf{X_s}$.

Next, we define a family of probability densities for each fibers of $h_{\sigma^2}$. Specifically, we define a density over $\mathbf{X_s}$ by $q : \mathbf{X_s} \to \mathbb{R}$ via $q(x_s) = \frac{p(x_s)}{\int_{X_s} p(x)dx}$. For $x \notin \mathbf{X_s}$, $q(x) = 0$. This allows us to define the domain restricted regression error

$$\text{err}_{\text{reg}}(h, s) = \int_{\mathbf{X_s}} q(x_s) \left\| h_\mu(x_s) - f(x_s) \right\|_2^2 dx_s. \tag{65}$$

Denote by $q(\mathbf{G}\mathbf{x_s}) = \int_{z \in \mathbf{G}\mathbf{x_s}} q(z)dz$ the probability of the orbit $\mathbf{G}\mathbf{x_s}$ on $\mathbf{X_s}$. Denote by $q_{\text{norm}}(z) = \frac{q(z)}{q(\mathbf{G}\mathbf{x_s})}$ the normalized probability density on the orbit $\mathbf{G}\mathbf{x_s}$ such that $\int_{\mathbf{G}\mathbf{x_s}} q_{\text{norm}}(z)dz = 1$. Let $\mathbb{E}_{\mathbf{G}\mathbf{x_s}}[f]$ be the mean of the function $f$ on the orbit $\mathbf{G}\mathbf{x_s}$ and let $\mathbb{V}_{\mathbf{G}\mathbf{x_s}}[f]$ be the variance of $f$ on the orbit $\mathbf{G}\mathbf{x_s}$,

$$\mathbb{E}_{\mathbf{G}\mathbf{x_s}}[f] = \int_{\mathbf{G}\mathbf{x_s}} q_{\text{norm}}(z) f(z) dz = \frac{\int_{\mathbf{G}\mathbf{x_s}} q(z) f(z) dz}{\int_{\mathbf{G}\mathbf{x_s}} q(z) dz} \tag{66}$$

$$\mathbb{V}_{\mathbf{G}\mathbf{x_s}}[f] = \int_{\mathbf{G}\mathbf{x_s}} q_{\text{norm}}(x) \left\| \mathbb{E}_{\mathbf{G}\mathbf{x_s}}[f] - f(z) \right\|_2^2 dz. \tag{67}$$

These definitions arise from the error lower bound on invariant regression in Wang et al. (2024) (see also Appendix 11.2.2). Intuitively, if a model $h$ is constant on an orbit where $f$ is varying, then the constant that minimizes the regression error is the average of $f$ on the orbit, and the induced error is the variance of $f$.

Finally, we define a generalization of the expected normalized calibration error (ENCE), as given by Eqn. 8 in Levi et al. (2022a). The main drawbacks of their ENCE metric are two-fold: it is defined in terms of binning approximations and assumes that $h_\mu(x)$ and $h_{\sigma^2}(x)$ are scalar values. Our formalism not only works for vectors and avoids binning, allowing for a discussion of continuous group symmetries. While binning approximations are still necessary to compute ENCE in practice, our theory supports the more generalized continuous case. We take the absolute value function applied to a vector $|\cdot|$ to be applied element-wise. Similarly, if we use $\sqrt{c \cdot s}$ as a vector, that means the square root function was applied element-wise to the vector $s$ after and scaled by a constant $c$. We will also note that vectors have a partial ordering where $\vec{a} \leq \vec{b}$ if $a_i \leq b_i$ for all $i$. Let $\mathbf{D}$ be the region of vectors $d$ bounded by $s_1 \leq d \leq s_2$. We define a probability density $r : \boldsymbol{\mathcal{S}} \to \mathbb{R}$ such that $\mathbb{P}(h_{\sigma^2}(x) \in \mathbf{D}) = \mathbb{P}(s \in \mathbf{D}) = \int_{\mathbf{D}} r(s)ds$. This is the push-forward of the density of $p$ over $h_{\sigma^2}$, equivalently, $r = h_{\sigma^2} \# (p)$.

**Generalized Expected Normalized Calibration Error.** We now have the necessary machinery to define our novel generalized ENCE metric that applies to vector-valued function. The goal for our learning task is that $h_\mu$ fits the function $f$ *and* $h_{\sigma^2}$ properly predicts confidence by minimizing our generalized ENCE metric (Equation 68):

**Definition 7** (Expected Normalized Calibration Error)**.** The ENCE penalizes the fiber-wise discrepancies between the error and the uncertainty averaged over all variances.

$$\text{ENCE} = \int_{\boldsymbol{\mathcal{S}}} r(s) \cdot \frac{\mathbb{E}_{X_s}\left[ \left\| \sqrt{\frac{2}{\pi}} s - |h_\mu(x) - f(x)| \right\|_2^2 \,\middle|\, h_{\sigma^2}(x) = s \right]}{\left\| \sqrt{\frac{2}{\pi}} s \right\|_2^2} ds. \tag{68}$$

We assume that the model class $\{h\}$ is arbitrarily expressive except that it is constrained to be invariant with respect to a group $\mathbf{G}$. That is to say, it is a universal approximator for compactly supported $G$-invariant functions.

**Remark 3.** If errors are normally distributed and a model is well calibrated, then $\sqrt{\frac{2}{\pi}}\sigma = \overline{|\mu - f(x)|}$ is the same as $\sigma^2 = \overline{(\mu - f(x))^2}$, where the factor of $\sqrt{\frac{2}{\pi}}$ comes from Geary (1935). We choose the former as a componenet in our metric because it allows us to later apply a theorem from Wang et al. (2024). Even though it appears slightly different than the binned approximate form in Levi et al. (2022a) it is functionally the same as it penalizes miscalibration in the same way.

We also discuss a variation of this problem where mean and variance predictions obey different $G-$equivariances in Appendix B.

We now state our main theorem for this section, which bounds the expected normalized calibration error for $G-$invariant functions and discusses the special case of minimized regression error using the bounds from Wang et al. (2024). Our bound can be used as a heuristic when the range of the regression error is known to interpret how well a calibrated a model is compared to the worst case scenario.

**Theorem 4.** If a model $h$ is $G-$invariant, then ENCE given by Equation 68 is bounded below by 0 and bounded above by $1 + \mathop{\mathbb{E}}_{\boldsymbol{\mathcal{S}}}\left[\frac{\text{err}_{\text{reg}}(h,s)}{\|\sqrt{\frac{2}{\pi}}s\|_2^2}\right]$. If $\text{err}_{\text{reg}}(h,s)$ is minimized, then the ENCE upper bound becomes $1 + \mathop{\mathbb{E}}_{\boldsymbol{\mathcal{S}}}\left[\frac{\int_{F_s} q(Gx_s)\mathbb{V}_{Gx_s}[f]dx_s}{\|\sqrt{\frac{2}{\pi}}s\|_2^2}\right]$.

*Proof.* We begin by observing that the entries in $\sqrt{\frac{2}{\pi}}s$ and $|h_\mu(x) - f(x)|$ are nonnegative. Therefore, we have that

$$0 \leq \mathbb{E}_{X_s}\left[\left\|\sqrt{\frac{2}{\pi}}s - |h_\mu(x) - f(x)|\right\|_2^2 \;\middle|\; h_{\sigma^2}(x) = s\right] \leq \mathbb{E}_{X_s}\left[\left\|\sqrt{\frac{2}{\pi}}s\right\|_2^2 + \left\||h_\mu(x) - f(x)|\right\|_2^2 \;\middle|\; h_{\sigma^2}(x) = s\right]. \tag{69}$$

Consequently,

$$\begin{aligned}
0 \leq \text{ENCE} &\leq \int_{\boldsymbol{\mathcal{S}}} r(s) \cdot \frac{\mathbb{E}_{X_s}\left[\left\|\sqrt{\frac{2}{\pi}}s\right\|_2^2 + \left\||h_\mu(x) - f(x)|\right\|_2^2 \;\middle|\; h_{\sigma^2}(x) = s\right]}{\left\|\sqrt{\frac{2}{\pi}}s\right\|_2^2} ds \\
&= \int_{\boldsymbol{\mathcal{S}}} r(s) \cdot \frac{\left\|\sqrt{\frac{2}{\pi}}s\right\|_2^2 + \mathbb{E}_{X_s}\left[\left\||h_\mu(x) - f(x)|\right\|_2^2 \;\middle|\; h_{\sigma^2}(x) = s\right]}{\left\|\sqrt{\frac{2}{\pi}}s\right\|_2^2} ds \\
&= \int_{\boldsymbol{\mathcal{S}}} r(s) ds + \int_{\boldsymbol{\mathcal{S}}} r(s) \cdot \left[\frac{\int_{X_s} q(x_s)\left\|h_\mu(x) - f(x)\right\|_2^2 dx}{\left\|\sqrt{\frac{2}{\pi}}s\right\|_2^2}\right] ds.
\end{aligned}$$

By the definition of domain restricted regression error in our problem setup, we conclude

$$\int_{X_s} q(x_s)\left\|h_\mu(x_s) - f(x_s)\right\|_2^2 dx_s = \text{err}_{\text{reg}}(h,s).$$

This gives us

$$\text{ENCE} \leq \int_{\boldsymbol{\mathcal{S}}} r(s)\left[1 + \frac{\text{err}_{\text{reg}}(h,s)}{\|\sqrt{\frac{2}{\pi}}s\|_2^2}\right] ds \tag{70}$$

$$= 1 + \mathop{\mathbb{E}}_{\boldsymbol{\mathcal{S}}}\left[\frac{\text{err}_{\text{reg}}(h,s)}{\|\sqrt{\frac{2}{\pi}}s\|_2^2}\right]. \tag{71}$$

Now, if the domain restricted regression error is minimized, then by Theorem 1 we have that

$$\text{err}_{\text{reg}}(h, s) = \int_{F_s} q(Gx_s)\mathbb{V}_{Gx_s}[f]dx_s \tag{72}$$

which completes the proof. □

**Remark 4.** A key part of the proof is that the level sets $X_s$ are closed under the action of $G$ by assumption that $h_{\sigma^2}(x_s)$ is $G$-invariant.

**Remark 5.** The upper bound in Equation 71 is not guaranteed to converge.

In Appendix C, we discuss a class of $G-$invariant functions where the approximation error can be made arbitarily small on each orbit.

It is difficult to compute the upper bound in practice without prior knowledge on the density of the input data. Despite this, we can consider special cases where the bound can be computed. Let us exemplify.

**Example 1.** The goal of this example is to show how the bound can be computed in a toy setting with realistic group symmetries that we may later study on a real data set. In this example, we consider a set $X$ of five chemical compounds and a codomain $Y$ of spectra. We will calculate the ENCE upper bound assuming a minimized regression error. Each molecule $x \in X$ has information related to position and atomic number. We assume that the model class $h$ has $E(2)$-invariance to the atomic positions, where $E(2)$ is the Euclidean group for $\mathbb{R}^2$. The atomic positions are shown in Figure 15. Note that for illustration purposes in this example, we are not giving the exact physically accurate atomic positions as you would find in Ramakrishnan et al. (2014) or a similar dataset. The dataset contains some duplicates of the chemical compounds up to transformations in $E(2)$. That is, the atomic numbers are the same but the atomic positions may be rotated and translated about the origin. Different molecule orientations are notated with $+$ and $\times$.

The input space $X$ contains molecules $\{CH_4(+), CH_4(\times), H_2O, SO_2, NH_3\}$. The map $f : X \to Y$ produces four unique spectra, one for each of the unique molecules. The model class $h$ predicts three unique spectra, as it produces the same spectra for both $H_2O$ and $SO_2$ due to their equivalence up to $E(2)$ in our simplified example. The model $h$ produces two distinct variance vectors – it predicts variance $s_1$ for the first two orbits and variance $s_2$ on the last orbit. The probability of obtaining each of the molecules is $\{0.125, 0.125, 0.125, 0.125, 0.5\}$ for $\{CH_4(+), CH_4(\times), H_2O, SO_2, NH_3\}$ respectively.

Along each row, Figure 15 shows an orbit of the representative set of $X$ in the first two columns, the spectral lines from $f$ and $h_\mu$ in the third column, and the associated variance $s$ from $h_{\sigma^2}(x)$ in the last column. Each molecule is titled with the name of the compound and the probability of sampling it in $X$. Consider the first row. Since $h$ is $E(2)$ invariant, $h$ is able to fully fit the function $f$ on the orbit containing two rotated version of $CH_4$. Thus, the regression error is zero. Along the second row, $h$ produces the same spectral lines for $H_2O$ and $SO_2$ since they are the same up to $E(2)$ in our example. Since $H_2O$ and $SO_2$ are equally probable, the output of $h$ that minimizes the regression error is just the average of the two spectral lines.

We now refer to Eqn. 72 to compute the domain restricted regression error for $s_1$. The first orbit gave us a regression error of zero. As such, we must have $\mathbb{V}_{Gx_s}[f] = 0$ on this orbit. This is certainly true, as $f$ is invariant to the atomic numbers, which are the same for the various rotations of methane. For the second orbit, we compute

$$\begin{aligned} q(Gx_s) &= \frac{p(H_2O) + p(SO_2)}{p(CH_4(\times)) + p(CH_4(+)) + p(H_2O) + p(SO_2)} \\ &= \frac{0.125 + 0.125}{0.125 + 0.125 + 0.125 + 0.125} \\ &= 0.5. \end{aligned}$$

In other words, when restricted to the domain that outputs a variance of $s_1$, we have a 50% chance of being on the orbit containing $H_2O$ and $SO_2$. The mean of the function $f$ on the orbit $Gx_s$ is just the average of the two spectral lines since the probabilities of $H_2O$ and $SO_2$ are equal. The variance can be computed as the average distance of the real spectral lines from the prediction $h$, which is itself just the average of the two lines. Obtaining exact values for the spectra from Ramakrishnan et al. (2014) and Coblentz Society

(1977), we can calculate a value of approximately 38.5 for the variance of $f$ on the second orbit. Thus, we can compute $\text{err}_{\text{reg}}(h, s_1) = (0.5 \times 0) + (0.5 \times 38.5) = 19.25$.

As with the first orbit, the minimizing regression error on the third orbit is zero. Since there is only one element in $X$ we can fit, the mapping between the molecule and the spectra is unique and can be made to fit the ground truth. So, $\text{err}_{\text{reg}}(h, s_2) = 0$.

Notice that the probability of obtaining $s_1$ is the same as the probability of obtaining $s_2$. Putting it all together, we get

$$\begin{aligned}\text{ENCE} &\leq 1 + \left(0.5 \cdot \frac{0}{||\sqrt{\frac{2}{\pi}s_2}||_2^2}\right) + \left(0.5 \cdot \frac{19.5}{||\sqrt{\frac{2}{\pi}s_1}||_2^2}\right) \\ &= 1 + \frac{9.625}{||\sqrt{\frac{2}{\pi}s_1}||_2^2}.\end{aligned}$$

We conclude that the upper bound for ENCE in our example is $1 + \frac{9.625}{||\sqrt{\frac{2}{\pi}s_1}||_2^2}$. This is plotted as a function of $||\sqrt{\frac{2}{\pi}s_1}||_2^2$ in Figure 16. We can see that in the limit as $||\sqrt{\frac{2}{\pi}s_1}||_2^2$ goes to infinity the upper bound on ENCE becomes 1. Alternatively, in the limit as $||\sqrt{\frac{2}{\pi}s_1}||_2^2$ goes to 0, the upper bound on ENCE diverges. The interpretation of the latter is that if the model is extremely confident, then any deviations from the mean prediction represents similarly extreme miscalibration.

### 12.3 Equivariant Regression Upper Bounds

Finally, we conclude with bounds on calibration error for equivariant functions. While our findings largely carry over from our discussion of invariant functions, we make subtle tweaks to our proof in order to compensate for the fact that elements in each fiber are no longer closed under the action of the group.

**Equivariant Regression Problem Setup.** Our setup is the same as for invariant regression, with the addition of the following: We define a matrix $Q_{\mathbf{Gx}} \in \mathbb{R}^{n \times n}$, $Q^\dagger(g\vec{x}) \in \mathbb{R}^{n \times n}$ such that $\int_{\mathbf{G}} Q^\dagger(g\vec{x})dg = \text{Id}$ via

$$Q_{\mathbf{Gx}} = \int_{\mathbf{G}} q(gx)\rho_Y(g)^T\rho_Y(g)\alpha(x, g)dg \tag{73}$$

$$Q^\dagger(gx) = Q_{\mathbf{Gx}}^{-1} q(gx)\rho_Y(g)^T\rho_Y(g)\alpha(x, g). \tag{74}$$

We also define $\mathcal{E}_{\mathbf{G}}[f, x]$ by

$$\mathcal{E}_{\mathbf{G}}[f, x] = \int_G Q^\dagger(gx)g^{-1}f(gx)dg. \tag{75}$$

**Theorem 5.** We assume the model class $\{h\}$ is equivariant. ENCE (Eqn. 68) is bounded below by 0 and bounded above by $1 + \mathop{\mathbb{E}}_{\boldsymbol{\mathcal{S}}}\left[\frac{\text{err}_{\text{reg}}(h,s)}{\|\sqrt{\frac{2}{\pi}s}\|_2^2}\right]$. If $\text{err}_{\text{reg}}(h, s)$ is minimized then the upper bound becomes $1 + \mathop{\mathbb{E}}_{\boldsymbol{\mathcal{S}}}\left[\frac{\int_F \int_G q(gx)\|f(gx) - g\mathcal{E}_G[f,x]\|_2^2 \alpha(x,g)dgdx}{\|\sqrt{\frac{2}{\pi}s}\|_2^2}\right]$.

*Proof.* Unlike before, we can not decompose an integral over $X_s$ into an iterated integral over $F_s$ and $Gx_s$. This is because without the assumption of invariance, the restriction $h_{\sigma^2}(x) = s$ no longer preserves entire orbits. Therefore, we instead note that we can rewrite the domain restricted regression error in terms of the original domain $X$. Consider the integral in Eqn. 65. Since $q(x) = 0$ for $h_{\sigma^2}(x) \neq s$, we may write

$$\text{err}_{\text{reg}}(h, s) = \int_X q(x)\left\|h_\mu(x) - f(x)\right\|_2^2 dx. \tag{76}$$

Said differently, the integrand takes on a value of 0 outside of $X_s$, so without loss of generality, we can replace $X_s$ with $X$ and $x_s$ with $x$ everywhere. Using the same argument as we did with invariant regression, we arrive at

$$\begin{aligned}\text{ENCE} &\leq \int_{\boldsymbol{\mathcal{S}}} r(s) \left[1 + \frac{\text{err}_{\text{reg}}(h, s)}{\|\sqrt{\frac{2}{\pi}} s\|_2^2}\right] ds \\ &= 1 + \mathop{\mathbb{E}}_{\boldsymbol{\mathcal{S}}} \left[\frac{\text{err}_{\text{reg}}(h, s)}{\|\sqrt{\frac{2}{\pi}} s\|_2^2}\right].\end{aligned}$$

Applying Theorem 2, if $\text{err}_{\text{reg}}(h, s)$ is a minimizer then we have

$$\text{ENCE} \leq 1 + \mathop{\mathbb{E}}_{\boldsymbol{\mathcal{S}}} \left[\frac{\int_F \int_G q(gx) \|f(gx) - g\mathcal{E}_G[f, x]\|_2^2 \alpha(x, g) dg dx}{\|\sqrt{\frac{2}{\pi}} s\|_2^2}\right]. \tag{77}$$

This completes the proof. □

**Remark 6.** The upper bounds presented here and in Section 12.2 are analogous to the maximum $\chi^2$ score one can attain averaged over all of the variances the model predicts.

### 12.4 Relation to Aleatoric Uncertainty

While our work in this section has explored the ramifications of equivariance on model miscalibration, our proofs did not reason about the type of uncertainty. Consider a model that attempts to disentangle aleatoric and epistemic uncertainty using evidential priors or similar negative log likelihood formulations. Evidential regression is far from perfect in its ability to disentangle these uncertainties (Valdenegro-Toro & Mori, 2022; Osband et al., 2023; Wimmer et al., 2023; Nevin et al., 2024; Jürgens et al., 2024). We define the *aleatoric bleed* with the goal of quantifying how far we are from accurately measuring uncertainties. While we may not always have access to such in practice, denote the ground truth aleatoric uncertainty at a given point $x$ by $f(x)$. We do not consider a ground truth epistemic uncertainty, as such a thing is not well defined. Now consider a model class $\{h : X \to \boldsymbol{\mathcal{S}}_{\text{aleatoric}}\}$. As before, $h$ is constrained to be equivariant. We insist that both $f$ and $h$ are nonnegative, but are otherwise arbitrarily expressive.

**Definition 8.** We define the aleatoric bleed as the regression error $\text{err}_{\text{reg}}(h)$ for a model $h$ that is attempting to predict the aleatoric uncertainty $\boldsymbol{\mathcal{S}}_{\text{aleatoric}}$.

We note that by Theorem 2, the aleatoric bleed has a known lower bound. Moreover, the aleatoric bleed can be easily computed in cases where the ground truth aleatoric uncertainty is identically the zero vector, which we will explore in §13.

## 13 Experiments

In this section, we conduct exploratory data analysis to better understand the relationship between equivariance and uncertainty. We study miscalibration on real and simulated datasets for both calibration and regression tasks. In particular, we study the effect of different levels of symmetry mismatch using the framework of correct, incorrect, and extrinsic equivariance (§13.1). Next, we examine the effects of equivariance under different levels of ground truth uncertainty (§13.2). We then study how different types of equivariance contribute to aleatoric bleed (§13.3). We follow by studying the disagregation of uncertainties into epistemic and aleatoric uncertainty, and dissect how equivariance plays a role in such (§13.4). Finally, we show experimentally that the bound on ENCE from §12.3 can be realized in a true experimental setting (§13.5-13.6). All experiments in this work were ran on a NVIDIA A100 GPU (Choquette et al., 2021) and all terminated in under 8 hours.

### 13.1 Swiss Roll and the Equivariance Taxonomy

Here, we study the effect of different notions of equivariance on ECE. Specifically, we use the taxonomy of correct, incorrect, and extrinsic equivariance, which we review in Appendix 11.2.1. Our study takes place on the vertically separated Swiss Roll distributions from (Wang et al., 2024). These distributions contain a $3D$ point cloud arranged in a spiral-like fashion, and binary labels are assigned to each point — see Figure 49 and Figures 7 and 11 in Wang et al. (2024). Specifically, we consider two different Swiss Roll distributions containing correct and incorrect equivariance. We build our dataset by taking samples from these distributions in different ratios. We then train an unconstrained MLP and a $z-$invariant network to predict the label. We provide further experimental details in Appendix E. Our aim is to realize a similar trend to accuracy in Wang et al. (2024) for ECE as a function of the correct vs. incorrect equivariance ratio used to build our dataset.

**Results.** While it may seem obvious that ECE is inversely correlated with model accuracy, we note this need not be the case. In fact, we will later see an example where it is not. Our experiment illustrates a circumstance wherein incorrect equivariance not only harms model accuracy, but also model calibration. When the correct ratio of equivariance is low, Figure 17 shows that not only will the model be correct less than about 70% of the time, but also its ECE will reach as high as 25%.

### 13.2 Galaxy Zoo Morphology and Trends in Group Order

The previous experiment suggested that a model's ECE improves as the amount of correct equivariance increases. Here, we seek to push the limits of just how far correct equivariance can help you. Specifically, we consider the task of galaxy morphology classification, which naturally has $E(2)-$invariance. We can approximate $E(2)-$invariance in CNNs with $C_n$ and $D_n$ group convolution layers, where $C_n$ denotes the cyclic group of order $n$ and $D_n$ denotes the dihedral group of order $n$. We examine trends in group order $n$. While any $C_n$ or $D_n$ has correct equivariance, an increase in $n$ certainly captures more of the underlying symmetry as you better approximate the $E(2)-$invariance (with the caveat of aliasing, see Zhang (2019); Karras et al. (2021); Gruver et al. (2023)). We look at trends in both accuracy and ECE under different levels of point-spread function (PSF) convolution. This allows us to look at performance under different levels of ground truth noise, which should ideally affect both the model confidence and accuracy. PSF blurring is also an extremely relevant source of noise for the astrophysics community, specifically weak lensing analysis and exoplanet imaging, see Appendix F.1. Our data comes from Galaxy Zoo images (Walmsley et al., 2024) of galaxies from the DESI and SDSS surveys. Further experimental details are recorded in Appendix F.

**Results.** Figure 18 shows that ECE does not follow the same trends in cyclic group order as accuracy. The result is in accordance with the fact that ECE has no absolute lower bound, but accuracy *does* have a lower bound that explicitly depends on the type of equivariance in your model. This is echoed in Appendix H, which shows similar trends for both cyclic and dihedral group order for both DESI and SDSS surveys. We note that the accuracy curves are reminiscent of Weiler & Cesa (2019); Pandya et al. (2023).

### 13.3 Vector Field Regression, Equivariance Taxonomy, and Aleatoric Bleed

**Vector Field Regression.** Our goal for this section is to demonstrate the relationship between the equivariance taxonomy and aleatoric bleed in a regression setting. We consider a model $h : \mathbb{R}^3 \to \mathbb{R}^3 \times \mathbb{R}^3$ that predicts two vector fields representing a mean and a variance prediction. That is, we predict vectors attributed to any given point in $\mathbb{R}^3$. We denote the ground truth vector field at a given point $x$ by $f(x)$, and $h$ is constrained to be equivariant with respect to $E(3)$. Figure 19 presents two examples of how the equivariance taxonomy can result in different levels of aleatoric bleed.

We consider two different ground truth functions $f$ designed to produce both correct and incorrect $E(3)-$equivariance such that we can explore how the equivariance taxonomy can affect downstream aleatoric bleed. ***1. Spiral.*** Let $Q$ be a 90 degree rotation matrix in $\mathbb{R}^3$. We define $f(x) := Qx$ . ***2. Sinusoidal.*** $f(x) := -\sin^2(||x||)x$. See that the spiral creates incorrect and extrinsic equivariance, since in general rotations in $\mathbb{R}^3$ do not commute. Therefore $f(gx) = Qgx$ and $g(f(x)) = gQx$ and $f(gx) \neq gf(x)$. For the

sinusoidal case, we note that rotations, translations, and reflections in $\mathbb{R}^3$ preserve the norm of a vector $x$. We have that $f(gx) = -\sin^2(||gx||)gx = -\sin^2(||x||)gx$ and $g(f(x)) = g(-\sin^2(||x||)x) = -\sin^2(||x||)gx$. Therefore $gf(x) = f(gx)$, and we have correct equivariance. In both cases, $f$ is completely deterministic, meaning any nonzero norm of a variance vector is indicative of aleatoric bleed.

For simplicity and visualization purposes, we construct our dataset with vectors in $\mathbb{R}^3$ with a $z$ component of 0 and we choose rotation matrices $Q$ that keep the vectors in the $xy-$plane for the spiral setup. We provide relevant training details in Appendix D.

**Results.** As expected, Figure 19 shows that the incorrect and extrinsic equivariance makes the $E(3)-$equivariant model unable to fit the data appropriately with its mean predictions. Consequently, it predicts extremely high variance vectors, as our $\beta-$NLL loss function can reach a local minimum when the variance prediction is significantly larger than the mean squared error. For details on the loss function and training setup, we remind the reader to see Appendix D. See in Figure 20 that despite the mean vector field failing to appropriately fit the data, the $\beta-$NLL loss is still fairly close to the MLP for vectors at any given angle $\theta$. However, in the case of the correct equivariance with the sinusoidal dataset, the correctly applied $E(3)-$equivariance helps the model both in terms of MSE and $\beta-$NLL, accurately fitting the data and minimizng aleatoric bleed.

### 13.4 Chemical Properties and Aleatoric Bleed

The goal of this experiment is to assess whether a model's learned variance predictions are themselves reliable in a setting that is more realistic than the vector fields in Figure 19. That is, we ask if the model's confidence predictions are consistent with what the ground truth variance should be, and how equivariance can affect this. Our experiment is again motivated by the analyses of Valdenegro-Toro & Mori (2022); Osband et al. (2023); Wimmer et al. (2023); Nevin et al. (2024); Jürgens et al. (2024), which illustrate that a model can conflate the uncertainty in its weights with the real uncertainty of the data.

**Scalar-Valued Predictions.** Our scalar-valued property prediction tasks take as input chemical compounds sourced from QM9s (Ramakrishnan et al., 2014). We predict various chemical properties with two different message-passing graph neural networks, one non-equivariant baseline and one with $E(3)-$equivariance. Specifically, our non-equivariant baseline is the GIN model (Xu et al., 2018) and we compare with an $E(3)-$invariant model (Batzner et al., 2022), with implementations based on (Backenköhler et al., 2023). Both models are equipped with independent feed forward neural network decoder heads which are used to learn a four parameter family that characterizes a Student t distribution. This in turn gives us enough degrees of freedom to reconcile epistemic and aleatoric uncertainties as described in §11.3. Further experimental details are provided in Appendix G. Physically, the relationship between these scalar values and chemical compounds should be a deterministic process, and accordingly the ground truth aleatoric uncertainty should always be zero. Since we are dealing with scalar values, aleatoric bleed reduces from a norm to a simple average over the square of predicted uncertainties. Note that the goal with this experiment is not to train the models to optimal performance; in fact, having models that cannot perfectly generalize is useful for us to study models with non-trivial uncertainties. We strive instead to compare models with similar accuracy but potentially varying levels of aleatoric bleed.

**Scalar-Valued Results.** In contrast to Figure 19, which showed the dangers of incorrect and extrinsic equivariance on aleatoric bleeding, we find that correct equivariance has little impact on aleatoric bleeding. As a specific case study, consider the dipole moment prediction task shown in Figure 21. We see that the aleatoric bleed is nearly identical between the GIN and $E(3)-$invariant models, with no significant deviations in how errors are distributed. This result mirrors what we found for classification: correct equivariance does little to improve model calibration, but incorrect equivariance can punish a model all the same.

In Table 3, we compare the aleatoric bleed between the baseline and equivariant models when their accuracy is comparable, which we quantify as having a mean absolute error (MAE) within 0.25. The model with the lower aleatoric bleed seems to depend neither on the performance of the model nor the inclusion of

| Chemical Property | Unit | GIN MAE | $E(3)-$Invariant MAE | GIN AB | $E(3)-$Invariant AB |
|---|---|---|---|---|---|
| $\varepsilon_{LUMO}$ | eV | 0.4404 | 0.6710 | **0.0081** | 3.0288 |
| $\Delta\varepsilon$ | eV | 0.6877 | 0.7283 | **0.0013** | 3.0287 |
| $U_0$ | eV | 0.2563 | 0.0563 | 0.0333 | **0.0053** |
| $U$ | eV | 0.2558 | 0.0563 | 0.0323 | **0.0053** |
| $U_0^{\text{ATOM}}$ | eV | 0.1908 | 0.1458 | 0.0193 | **0.0052** |
| $G^{\text{ATOM}}$ | eV | 0.7954 | 0.7706 | **0.0000** | 0.0014 |
| $A$ | GHz | 0.0667 | 0.2504 | **0.0000** | 0.0014 |
| $B$ | GHz | 0.1625 | 0.0992 | 0.0066 | **0.0040** |
| $C$ | GHz | 0.0780 | 0.0493 | **0.0008** | 0.0013 |
| $\langle R^2 \rangle$ | $(a_0)^2$ | 0.8621 | 0.8956 | **0.0005** | 0.0013 |

Table 3: Accuracy and Aleatoric Bleed (AB) for various scalar properties in QM9s for baseline and equivariant graph neural network models. Predictions and error estimates are given for the $z-$scored scalar values.

equivariance. Our findings support the same conclusion that correct equivariance does little to help prevent aleatoric bleed.

In §12.4, we discussed how the aleatoric bleed has a known lower bound. The results from this experiment suggest that the lower bound is not tight enough to be meaningful to practitioners working on scalar-valued properties in the QM9 dataset. This is likely because our invariance constraint here is correct, so the lower bound on aleatoric bleed is zero. This is in contrast to the vector regression spiral experiment where incorrect equivariance clearly caused an increase of aleatoric bleed.

**Vector-Valued Predictions.** This experiment highlights a need for qualitative analysis to work in tandem with our aleatoric bleeding metric for high-dimensional outputs. In particular, aleatoric bleed fails to describe the individual coordinates in which the predicted variance vector suffers the most in terms of bleeding.

This experiment again uses QM9s, however, this time we instead predict spectral lines emitted from the chemical compounds using a network with steerable $E(3)-$vectors (Brandstetter et al., 2021). As before, we use independent feed forward neural network decoder heads in order to learn a four parameter family that characterizes a Student t distribution. Again, physics tells us that the ground truth aleatoric uncertainty should ideally be exactly zero. Any non-zero uncertainties are indicative of epistemic uncertainties bleeding into the aleatoric uncertainty prediction. As such, the aleatoric bleed becomes the mean squared norm of the predicted aleatoric variance vectors.

**Vector-Valued Results.** We compute an aleatoric bleed of $\approx 17.613$. However, as seen in Figure 22, the model tends to conflate high frequency signals with noise. The model's aleatoric uncertainty follows the epistemic uncertainty quite closely, indicating that the model can not tell them apart. This is echoed with 31 examples in the Appendix I. Thus, we conclude that the aleatoric bleed is only vivid enough to tell a practitioner that the model is confusing different sources of uncertainty. Evidently, the metric can not tell a practitioner *where* the model tends to mess up the uncertainty estimate without additional diagnostics. That is not to say the aleatoric bleed metric is not useful; rather, we suggest practitioners should proceed with caution when examining aleatoric bleed.

Additionally, while not our main contribution, we point out that our model's raw performance is comparable to the state-of-the-art DetaNet (Zou et al., 2023b), with both models achieving $R^2$ scores above 0.9 and often close to 1.0 on QM9s test molecules. Our steerable $E(3)$ vector-based model also provides initial steps towards quantifying its own uncertainty. We further discuss the merits and limitations of our approach compared to DetaNet in Appendix G. We leave further model development and evaluation as an oppurtunity for future work.

### 13.5 Chemical Properties and Binning Approximations for ENCE

**Binning Approximations.** A necessary prerequisite for attempting to realize our upper bound on ENCE as derived in §12.3 is to understand if we can well approximate Equation 68 with binning approximations. Specifically, we must create discrete bins for our output variance vectors so that we can approximate the continuous density $r(\sigma^2)$. While we need not use approximations to compute the theoretical upper bound on the calibration error term in Equation 68, the binning approximations for computing the density term are unavoidable (Guo et al., 2017). We motivate our binning approximations for vector-valued regression talks with the established formalism for scalar-valued variance prediction binning approximations. Levi et al. (2022a) employ binning approximations as follows: Assume that your model outputs a mean $\hat{y}$ and a standard deviation $\sigma$. First, divide samples $\sigma$ into $N$ equally spaced bins $B_j$ between the minimum and maximum $\sigma$ predictions. Group the model outputs according to the $\sigma$ bins. The following per bin quantities are then computed:

$$RMV(j) = \sqrt{\frac{1}{|B_j|}\sum_{t\in B_j}\sigma_t^2} \tag{78}$$

$$RMSE(j) = \sqrt{\frac{1}{|B_j|}\sum_{t\in B_j}(y_t-\hat{y}_t)} \tag{79}$$

$$ENCE = \frac{1}{N}\sum_{j=1}^{N}\frac{|RMV(j)-RMSE(j)|}{RMV(J))}. \tag{80}$$

For our generalized ENCE metric, we adopt a similar procedure for approximating with bins. If our variance vector is in $\mathbb{R}^m$, then we divide *each dimension* $m$ into $N$ bins $B_{n,m}$. For each dimension, the bins are equally sized and bounded by the minimum and maximum variance prediction in that coordinate. Each coordinate of a model's variance output will be in one of $N$ bins in each dimension $m$. There will therefore also be sequences of $m$ bins $B_j := \{B_i\}_{i=1}^m \subseteq B_{n,m}$ that contain entire vectors. If a variance vector falls into a bin sequence $B_j$, then we may index that vector with a subscript $t$ along with the corresponding error vector. Our approximation is thus as follows:

$$h_\sigma = \sqrt{h_{\sigma^2}} \tag{81}$$

$$e = |h_\mu - f| \tag{82}$$

$$MNS(j) = \frac{1}{|B_j|}\sum_{t\in B_j}\|\sqrt{\frac{2}{\pi}}h_{\sigma,t}\|_2^2 \tag{83}$$

$$MNCE(j) = \frac{1}{|B_j|}\sum_{t\in B_j}\|\sqrt{\frac{2}{\pi}}h_{\sigma,t}-e_t\|_2^2 \tag{84}$$

$$ENCE = \frac{1}{N}\sum_{j}|B_j|\frac{MNCE(j)}{MNS(j)} \tag{85}$$

where $MNS$ abbreviates mean norm standard deviation and $MNCE$ abbreviates mean norm calibration error. The fraction $\frac{|B_j|}{N}$ can be intuited as the term approximating the true probability density $r(\sigma^2)$.

The task of using bins to approximate ENCE is already difficult in the case of scalar values. As Pernot (2023) demonstrates, the ENCE metric for scalar valued predictions (Levi et al., 2022a) already exhibits an "annoying" correlation with the number of bins used. Specifically, for $N$ bins used, the ENCE grows linearly with $N^{1/2}$ for scalar-valued outputs.

We specifically compute calibration error in terms of the predicted epistemic uncertainty. Calibration error is intended to compare a model's own confidence with its error, making epistemic uncertainty the more suitable

thing to use. Additionally, given our discussions of aleatoric bleed, we find the epistemic uncertainty to be a more reliable metric here.

**Binning Approximation Results.** Intriguingly, we find that our generalized ENCE instead increases linearly with $N$ when applied to our chemical spectra results, as shown in Figure 23 . We study this for up to 100 bins, in accordance with Scalia et al. (2020), even though most other works use $\sim 10-15$ bins (e.g., Palmer et al., 2022; Busk et al., 2021; Levi et al., 2022a; Scalia et al., 2020). Our difference in scaling law can potentially be attributed to two factors. First, our metric penalizes differences in $\sim 0.8\times$ the standard deviation with mean absolute error instead of differences of variance with mean squared error. Second, this is a possible result of the curse of dimensionality. The probability that any two vectors occupy the same region in $\mathbb{R}^n$ is very small for large $n$, this in turn makes it hard to approximate the true density $r(s)$ in Equation 68. In this specific case we are in $\mathbb{R}^{3501}$, the resolution of the spectral line.

The consequence of this dependence on bin number $N$ has the following sequence: we can only use ENCE to inform us of relative model miscalibration. That is, we can use the metric to tell us if one model is more miscalibrated than another. However, due to this dependence on the number of bins $N$, we can not reliably conclude that a model is well calibrated simply because it has a low ENCE score.

### 13.6 Chemical Properties and Realizing the Upper Bound on ENCE

**Realizing the Bound.** In this experiment, we revisit the spectral line predictions from §13.4 and show that the computed ENCE respects the upper bound given in §12.3 *regardless* of the choice of bin number. In §13.5 , the number of bins considered was based on a range from previous works. Accordingly, the following calculations will show that no matter how liberal we are with binning, the upper bound still holds.

In Equation 71, we showed that the upper bound on ENCE is related to the domain restricted regression error. Since the model we are studying is specifically invariant, we are interested in the error on entire orbits that are in the pre-image of $h_{\sigma^2}^{-1}(\sigma^2)$. The QM9 dataset is canonicalized such that molecules are unique up to rotation. Therefore, we include 90, 180, and 270 degree rotations of each molecule in the test set. In this way, we can compute the upper bound on regression error on non-trivial orbits and in turn compute an upper bound on ENCE. [5] For the true label corresponding to the rotated molecules in the test set, we consider two variations, each corresponding to rows (1) and (2) in Figure 15. ***1. Correct Augmentation.*** Here, the labels for the rotated molecules are the same as the original. In this circumstance, our $E(3)-$invariance is correct, and we expect our ENCE to be well within the bound. Additionally, we expect that this data augmentation matches the underlying physics of our world. ***2. Incorrect Augmentation.*** Here, the spectral line has an explicit dependence on the rotation angle of the molecule. Specifically, each 90 rotation corresponds to a reflection over the $y-$axis. This augmentation is not true to real physics, and is done to make our data have incorrect $E(3)-$invariance. In this way, we can illustrate how having incorrect equivariance takes us closer to our theoretical upper bound.

To upper bound the regression error on each orbit, we put two clamps on our $E(3)-$invariant model, which we view as part of the function approximator $h$. Since our upper bound is given by $\mathop{\mathbb{E}}_{h_{\sigma^2}}\left[\frac{\text{err}_{\text{reg}}(h,s)}{||\sqrt{\frac{2}{\pi}}s||_2^2}\right]$, these clamps ensure that $\text{err}_{\text{reg}}(h,s) \not\to \infty$ and $||\sqrt{\frac{2}{\pi}}s||_2^2 \not\to 0$. In turn, we can compute an upperbound $ENCE < \infty$. The first clamp bounds the output to be between $-3.3$ and $3.3$ on each coordinate, which is roughly the minimum and maximum of the $\log_{10}$ intensities of the true spectra and is in accordance with the fact that we trained on the spectra normalized by a factor of $1/3.3$. This clamp is after the pooling layer that ensures invariance and thus does not cause the $E(3)-$invariance to break. With this clamp, we are able to determine the max error for each label. We also impose a clamp on the variance prediction. If the variance prediction goes towards the zero vector then the upper bound on ENCE will also grow without bound. Thus, we clamp the variance predictions such that the minimum the norm $||\sqrt{\frac{2}{\pi}}s||_2^2$ can be is $\approx 7.0$, which was roughly the minimum of the computed samples before we introduced a clamp. With these clamps and the approximation of the

[5] An important reason why this works is that the spherical harmonic embedding of coordinates in $\mathbb{R}^3$ is equivariant (Appendix A4. Brandstetter et al., 2021).

density $r(s)$, we are able to get compute the true upper bound on $h$ subject to the binning approximations needed to approximate $r(s)$.

**Realizing the Bound Results.** As seen in Figure 24, our computed ENCE is less than the predicted upper bound no matter how many bins are used to approximate the density $r(s)$. Since our model seems to converge fairly well to the true absorption spectra, it is to be expected that our result is well within the upper bound for correct augmentation. Similarly, incorrect augmentation introduces error into our results, causing bringing our computed ENCE closer to its upper bound.

## 14 Limitations

**The Potential for Biased Estimators.** ECE and ENCE are difficult to compute in practice due to discrete binning approximations (Pernot, 2023), as the lack of an unbiased estimator adds uncertainty as to how reliable computed ECE and ENCE scores are. In this work, we attempted to circumnavigate the lack of a known unbiased estimator in the following ways. First, in classification settings where the model is consistently high in its confidence, we increase the granularity at which we bin, see Appendix E and F. Second, we carefully studied the effect of binning approximations in §13.5 and used that as a form of ablation in §13.6.

**Single Label Metric.** ECE only computes miscalibration with respect to a single label, but does not consider secondary or tertiary outputs that may be useful to practitioners as done in (Nixon et al., 2019).

**Upper Bound Computation.** Our computation of the upper bound in §13.6 depended upon experiment-specific clamps on our model output that correspond with an understanding of the underlying physics, relevant inductive biases, and the dataset. As such, these clamps are not generalizable outside of this specific case study. We recognize that different experimental settings may require different constraints to realize the proven upper bounds.

## 15 Future Work

Having laid the groundwork for a first unified theory for equivariance and uncertainty, there are several interesting and exciting potential avenues for future work:

**Supremum.** Our upper bound for ENCE presented in §12.3 may not be the supremum, and we seek to try and tighten the bound such that it more closely follows the observed ENCE or prove that no tighter bound exists.

**Mixed equivariances.** Our theory on regression could potentially be extended to circumstances where the output mean and variance predictions obey different $G-$equivariances. We touch on this only briefly in Appendix B.

**Scaling Laws.** We seek to better understand the relationship between miscalibration, incorrect and extrinsic equivariance, and model scale. We intend to study these scaling laws with a study similar to Brehmer et al. (2024).

**Unbiased Estimator.** An unbiased estimator for ENCE that does not depend on binning approximations would be an extremely useful contribution, and we will study such estimators in forthcoming work.

**Robotics.** One experimental domain where our work is closely applicable that we will further investigate is robotics. While our work can be difficult to frame in typical Q-learning setups since the optimal solution to the Bellman update function is unique (c.f. Figure **??**), a discussion of both equivariance and uncertainty quantification lends itself naturally to behavior cloning tasks such as in Jia et al. (2023). Accordingly, future work will examine calibration error for equivariant models in an imitation learning environment.

**Cosmology.** Another closely related experimental domain is cosmological large-scale structure. In particular, we will benchmark calibration error with symmetry-preserving models using the benchmark released in Balla et al. (2024). The appeal of this benchmark is that it includes graph-level predictions on $\Lambda CDM$ (Ryden, 2016; Carroll, 2019) cosmological parameters $\Omega_m$ and $\sigma_8$. In cosmology, these predictions are more commonly phrased as constraints on posterior distributions rather than point estimates (e.g., Collaboration: et al., 2016; Abbott et al., 2022), and so this is a natural playground for our work.

## 16 Discussion and Conclusions

Experiments in the natural sciences, especially in data-sparse settings, ***demand*** both equivariance and uncertainty estimation, and yet, no general theory for explaining how equivariance relates to uncertainty exists in the literature. We fill this gap, presenting the first unified theory explaining how equivariance relates to uncertainty estimation. We proved upper bounds on model calibration error for a class of functions that are arbitrarily expressive except that they are constrained to be equivariant. We do this in both classification and regression settings. This leads onto a set of experiments illustrating how the proven bounds manifest in practice.

Our core takeaways are best explained in light of the equivariance taxonomy. We highlight how incorrect and extrinsic equivariance can provably degrade model calibration for both classification and regression tasks. Our experiments support this. In the cases of the swiss roll and vector field regression experiments, both incorrect and extrinsic equivariance not only make the model less accurate, but also more poorly calibrated. In the case of the vector field regression experiment we also saw that incorrect and extrinsic equivariance contributed to aleatoric bleed. By contrast, we have shown that a model with correct equivariance is not necessarily better calibrated than a similarly sized non-equivariant baseline. As illustrated by the galaxy morphology classification and scalar-valued chemical property prediction experiments, equivariance is not strong enough to help a model become well calibrated nor is it strong enough to prevent aleatoric bleeding. It was only for the vector regression experiment on the sinusoidal dataset that the introduction of correct equivariance was able to significantly improve the raw performance and prevent aleatoric bleeding. The vector regression result is especially interesting in light of the galaxy morphology classification experiment, which showed that correct equivariance can help a model perform better without necessarily making it better calibrated.

## 17 Reproducability Statement

Our codebase containing all of our experiments, as well as the instructions to reproduce our results, is publicly available at .

We provide further experimental details and sources for the datasets used in this work throughout Appendices D, E, F, and G. We'd like to highlight the work of Wang et al. (2024) and Pandya et al. (2025a); the artifacts associated with these two works were simple to reproduce and aided us in our study.

## 18 Ethics Statement

Authors have no conflicts of interest to disclose.

### Acknowledgments

E.B. and J.G. acknowledge feedback and overwhelming support from students in Northeastern University's Mathematics course, `Research Capstone` (MATH4020), wherein this work was completed. In particular, we would like to thank Zachary Eisbach for helpful discussions. E.B. additionally thanks the Biomarkers team at AstroAI (Garraffo, 2024) for helpful discourse. E.B. and J.G. thank the Little Debbie company for providing 12 Swiss Rolls for consumption at the Star Market located at 53 Huntington Ave, Boston, MA. E.B. thanks Professor Suciu, whose real analysis homework inspired Appendix C. Finally, E.B. and J.G. thank members of the Geometric Learning Lab for helpful discussions and welcoming support.

# Part IV

# Is it *really* a planet? Coupling Evidential Regression with Differentiable Rendering for JWST NIRCam PSF Characterization and Exoplanet Imaging


## Abstract

Modeling point-spread function (PSF) aberrations using differential forward models of optical systems fails to account for the inherent uncertainties in photometry, sky noise, and "unknown unknowns" in the forward model. While accurate PSF models are crucial for all sorts of astronomical analysis, the goal of this work is to create PSF models for direct imaging of transiting exoplanets. In order to confirm an exoplanet planet discovery, we need a principled way to disambiguate planets from image artifacts such as hot pixels or dust. It is therefore of interest to quantify the uncertainty in our per-pixel predictions of the PSF model and whether we can leverage uncertainty quantification to produce a more reliable image of an exoplanet. In this work, we combine a differentiable forward model for the James Webb Space Telescope PSF with evidential regression so that we can model both the aleatoric and epistemic uncertainties of our PSF model. We coin our method EDDO-UQ, as an extension of the EDDO framework. We show the power of EDDO-UQ with clear direct detection of exoplanets using both simulated data and planet HIP 65426b.


## 19 Introduction

The James Webb Space Telescope Near Infrared Camera (JWST NIRCam) has unprecedented angular resolution, with pixel scales of $0.031''$/pixel and $0.063''$/pixel for the short and long wavelength channels, respectively. With JWST, we are now able to image not just galaxies and stars, but even things as small scale as individual planets. To put this into perspective, the exoplanet HIP 65426b was recently imaged by Carter et al. (2023). The planet is 356 light years away and has a radius roughly the size of Jupiter. With a corresponding angular size of roughly $6.4 \times 10^{-6}$ arcseconds and measuring using unresampled $F444W$ data, we can expect the planet to occupy $\frac{1}{1000}$ of a pixel on the detector. However, due to diffraction effects and small optical imperfections, as captured by the telescope point spread function (PSF), we are able to see the planetary light spread out over a few pixels!

The direct detection of HIP 65426b was accomplished computationally using pyKLIP (Wang et al., 2015). One of the core difficulties of direct exoplanet detection is that the light emanating from the transiting planet is of the same intensity of the speckle noise (Soummer et al., 2007) produced by the host star. pyKLIP attempts to model the star PSF in order to reveal the exoplanet behind the speckle noise.

Unfortunately, pyKLIP does not take into account the optical physics of JWST NIRCam, meaning it has the potential to make predictions that are non-physical. In response, subsequent works (e.g., Liaudat et al., 2023; Feng et al., 2025) have developed approaches to PSF characterization using differentiable optical models. These approaches optimize free parameters that are subject to the constraints imposed by the various instruments inside the telescope.

While these works have made great strides towards physically accurate PSF models, it can still be unclear when the PSF model is "good enough." That is to say, when subtracting off the effect of the PSF on a science image, it can be unclear whether the remaining artifact is truly an exoplanet or simply a combination of shot noise, read noise, and model misspecification (e.g., Golomb et al., 2021). The objective of this work is therefore to couple uncertainty quantification with differentiable rendering methods, which enables us to

embed inductive biases from optical physics into our modeling while also modeling uncertainty. We take an approach rooted in evidential regression, which gives us a precise way of identifying the per-pixel uncertainties in our PSF model due to both model mispecification and detector noise. We coin our method Exoplanet Detection with Differentiable Optics and Uncertainty Quantification (EDDO-UQ), as it extends the EDDO framework described in Feng et al. (2025). We show that EDDO-UQ is able to produce reliable uncertainty estimates of the PSF model on both real and simulated data, and furthermore that the uncertainty estimates themselves can be leveraged to make exoplanet detections easier. The remainder of this paper is organized as follows. In §20, we describe the methods of differentiable rendering and uncertainty quantification used in this work. We show the results of our method in §30. We offer conclusions in §31.

## 20 Method

**Evidential Regression.** Our primary method is the use of deep evidential regression to simultaneously model the per-pixel data (aleatoric) and model (epistemic) uncertainties[6] using the the formalism provided in Amini et al. (2020). Modeling of epistemic uncertainties is a core contribution of our method, since estimates of aleatoric (usually photometric) uncertainties often come with the JWST data products. However, the supplied aleatoric uncertainties can still be unreliable due to things like dust (Saydjari & Finkbeiner, 2022). It is therefore advantageous to have a model like EDDO-UQ that simultaneously estimates both.

We say that our data points $(y_1, \ldots, y_n) \sim \mathcal{N}(\mu, \sigma^2)$ and furthermore $\mu$ and $\sigma^2$ have priors

$$\begin{aligned} \mu &\sim \mathcal{N}(\gamma, \sigma^2 \nu^{-1}) \\ \sigma^2 &\sim \Gamma^{-1}(\alpha, \beta) \end{aligned}$$

where $\Gamma(\cdot)$ is the gamma function. Let $m = (\gamma, \nu, \alpha, \beta)$, and $\gamma \in \mathbb{R}$, $\nu > 0$, $\alpha > 1$, $\beta > 0$. One can then show that $p(y_i|m) = St(y_i; \gamma, \frac{\beta(1+\nu)}{\alpha\nu}, 2\alpha)$, where the St distribution is given by

$$St(t; \mu, \sigma, \nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu}\sigma\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{1}{\nu}\left(\frac{t-\mu}{\sigma}\right)^2\right)^{-(\nu+1)/2}.$$

Characterizing the St distribution as a four-parameter family is useful because it allows us to define prediction, aleatoric uncertainty, and epistemic uncertainty in a rigorous way:

$$\begin{aligned} \mathbb{E}[\mu] &= \gamma \quad \text{(Prediction)}, \\ \mathbb{E}[\sigma^2] &= \frac{\beta}{\alpha - 1} \quad \text{(Aleatoric Uncertainty)}, \\ \text{Var}[\mu] &= \frac{\beta}{\nu(\alpha - 1)} \quad \text{(Epistemic Uncertainty)}. \end{aligned}$$

In evidential regression, the goal is to learn the four-parameter family $m$ that best describes the data $(y_1, \ldots, y_n)$ by minimizing the negative log likelihood $-\log p(y_i|m)$. This in turn gives us our prediction, aleatoric uncertainty, and epistemic uncertainty from a single training run.

**Differentiable Forward Modeling.** We use the differentiable forward model for the James Webb Space Telescope Near Infrared Camera (JWST NIRCam) described in Feng et al. (2025) for our prediction $\gamma$. Our differentiable renderer $\mathcal{G}$ takes as input the position of the guide star $p_{\text{star}} \in \mathbb{R}^2$ and a wavefront aberration map $\phi \in \mathbb{R}^{H_\phi \times W_\phi}$. $\mathcal{G}$ models four operations extracted from WebbPSF specifications (Perrin et al., 2014b) in order to model NIRCam: We have the pupil entrance function $\mathcal{P}$, the focal plane mask $\mathcal{B}$, the Lyot Stop $\mathcal{C}$,

[6] Some fields refer to these as statistical and systematic uncertainties

and the NIRCam instrument $\mathcal{D}$. We will additionally use $\mathcal{F}$ to denote the Fourier transform. Our estimate of the science image is then

$$\hat{\gamma} = |\mathcal{F}D \cdot \mathcal{C} \cdot \mathcal{F}\mathcal{B} \cdot \mathcal{F}\mathcal{P} \cdot (x^{(0)} \cdot e^{ip_{\text{star}}} \cdot e^{i\phi})|^2$$

where $x^{(0)}$ is an input complex wavefront, typically a plane wave. For the purposes of this work, $\phi$ will be a grid of $1024 \times 1024$ learnable parameters.

**Solving for the PSF.** The negative log likelihood loss for $p(y_i|m)$ is known to be

$$\begin{aligned}
\mathcal{L}_i^{\text{NLL}}(w) &= \frac{1}{2}\log\left(\frac{\pi}{\nu}\right) - \alpha\log(2\beta(1+\nu)) \\
&+ \left(\alpha + \frac{1}{2}\right)\log((y_i - \gamma)^2\nu + 2\beta(1+\nu)) \\
&+ \log\left(\frac{\Gamma(\alpha)}{\Gamma(\alpha+\frac{1}{2})}\right).
\end{aligned}$$

We couple our negative log likelihood loss with a standard $L1$ regularization loss, which was shown to help training stability in Amini et al. (2020). This gives us

$$\begin{aligned}
\mathcal{L}_i^{\text{R}}(w) &= |y_i - \mathbb{E}[\mu_i]| \cdot \Phi \\
&= |y_i - \gamma| \cdot (2\nu + \alpha) \\
\mathcal{L}_i(w) &= \mathcal{L}_i^{\text{NLL}}(w) + \mathcal{L}_i^{\text{R}}(w).
\end{aligned}$$

We approximate $\hat{\gamma}$ with our differentiable renderer $\mathcal{G}$; however, our estimates for $\hat{\alpha}, \hat{\beta}, \hat{\nu}$ are almost entirely unconstrained. The estimates are modeled as a grid of learnable parameters the size of our predicted science image. We apply the softplus function to each and add 1 to $\hat{\alpha}$ to enforce the restrictions $\hat{\nu} > 0$, $\hat{\alpha} > 1$, $\hat{\beta} > 0$.

Once we have our trained model, we look for exoplanets in the residual of the image $y$ and the PSF prediction $\gamma$. We can study the signal-to-noise with $\chi^2$ images

$$\begin{aligned}
\chi^2_{\text{al}} &= \frac{(y - \hat{\gamma})^2}{\left(\frac{\hat{\beta}}{\hat{\alpha}-1}\right)} \\
\chi^2_{\text{ep}} &= \frac{(y - \hat{\gamma})^2}{\left(\frac{\hat{\beta}}{\hat{\nu}(\hat{\alpha}-1)}\right)}.
\end{aligned}$$

To quantify the performance of EDDO-UQ, we set up a series of experiments featuring both simulated and real images of an exoplanet near a bright star. Our simulated data consists of 10 mock realizations of the planet-star system with controlled shot and read noise added to each realization. The initial star and wavefront abberations are rendered using an untrained EDDO and a hidden planet is manually added behind one of the speckles. Real images of the exoplanet HIP 65426b (Carter et al., 2023) are sourced directly from the Mikulski Archive for Space Telescopes we use data taken from MAST[7]. In our experiments, we train our model with a Cosine learning rate scheduler and AdamW optimizer. For simulated data we use 3000 epochs and for real data 6000 epochs.

[7] https://webbtelescope.org/contents/media/images/01GBT1E93YV7YND5MFS1603FWJ

## 21 Results

**Simulated Data.** This experiment confirmed that EDDO-UQ can identify an exoplanet in a simulated data setting. Interestingly, Figure 25 shows a spike in both uncertainty maps where the synthetic exoplanet is. This is unsurprising, as the renderer $\mathcal{G}$ is meant to model speckle noise from the center guide star and not any other object. However, we are reassured by the fact that the uncertainty is still the same order of magnitude as the minimum, or in other words, that the uncertainty does not dominate the signal. Also of note is that the epistemic uncertainty less than the aleatoric uncertainty, which indicates that the inability to fit the data is bounded more by the read noise and shot noise than the expressivity of the model.

**Exoplanet HIP 65426b.** In this experiment, we establish that EDDO-UQ can directly detect exoplanets and that uncertainty estimation can help identify the planet. We study planet HIP 65426b (Carter et al., 2023) because it was recently imaged using the pyKLIP package (Wang et al., 2015), giving us a baseline to compare with. Notably, our planet location is consistent with what was found in Carter et al. (2023).

As seen in Figure 26, EDDO-UQ is able to reveal exoplanet HIP 65426b alongside its estimates for the epistemic and aleatoric uncertainties. Again, we find that the epistemic uncertainty is smaller than than the aleatoric uncertainty. When we weight the squared residuals by these uncertainties, as we do in Figure 27, we find that the hidden exoplanet becomes even more obvious. Put differently, our exoplanet signal is much higher than the noise, but very little of the surrounding pixels contain any signal compared to the noise.

The epistemic and aleatoric uncertainties are both highest around the center of the image. This is likely in accordance with the fact that the majority of the shot noise is in the center where the star shines most intensely, even if the coronagraph tries to mitigate that effect.

## 22 Discussion and Conclusions

In this work, we developed EDDO-UQ, a framework that couples uncertainty quantification with exoplanet detection with differentiable optics. We tested EDDO-UQ on both real and simulated data sets and demonstrated that by incorporating uncertainty estimation into our model, we can produce better and more contextualized science images. Our success can be partially explained by the results of Seitzer et al. (2022). They show that including uncertainty parameters in the model causes the gradients of the loss function to preferentially impact the pixels where the uncertainty is low. Therefore, during the optimization loop, the parameters related to the pixels in the science image corresponding to the core aberrations of the PSF are optimized more than the ones corresponding to background noise. This enables us to simultaneously learn a descriptive PSF model while also fitting uncertainty parameters.

More broadly, our work enables a more rigorous evaluation of candidate exoplanets as it allows users to simultaneously diagnose both modeling uncertainties and photometric uncertainties when fitting for the PSF. In turn, we can weight the final exoplanet image by the uncertainties and identify if our signal really dominates the noise. The epistemic uncertainties in particular provide an added measure of how reliable the PSF fit is, and the aleatoric uncertainties are useful in circumstances where the photometric measurements are unreliable. Our work also enables likelihood analysis of observing a pixel under the learned St distribution, paving the way for future automated exoplanet detection.

# Part V

# Efficient Point Spread Function Modeling with ShOpt.jl: A Point-spread Function Benchmarking Study with JWST NIRCam Imaging


## Abstract

With their high angular resolutions of 30–100 mas, large fields of view, and complex optical systems, imagers on next-generation optical/near-infrared space observatories, such as the Near-Infrared Camera (NIRCam) on the James Webb Space Telescope (JWST), present both new opportunities for science and also new challenges for empirical point spread function (PSF) characterization. In this context, we introduce `ShOpt`, a new PSF fitting tool developed in Julia and designed to bridge the advanced features of PIFF (PSFs in the Full Field of View) with the computational efficiency of PSFEx (PSF Extractor). Along with `ShOpt`, we propose a suite of non-parametric statistics suitable for evaluating PSF fit quality in space-based imaging. Our study benchmarks `ShOpt` against the established PSF fitters PSFEx and PIFF using real and simulated COSMOS-Web Survey imaging. We assess their respective PSF model fidelity with our proposed diagnostic statistics and investigate their computational efficiencies, focusing on their processing speed relative to the complexity and size of the PSF models. We find that `ShOpt` can already achieve PSF model fidelity comparable to PSFEx and PIFF while maintaining competitive processing speeds, constructing PSF models for large NIRCam mosaics within minutes.


## 23 Introduction

The inherent limitations of optical systems introduce artifacts into telescope imaging. Effects like diffraction, optical aberrations, atmospheric turbulence (if applicable), and telescope jitter are summarized in the telescope's point spread function (PSF): the response of the optical system to an idealized point of light. Good PSF modeling during image processing reduces the impact of things like atmospheric turbulence and optical aberrations during scientific analysis. Failure to model the PSF correctly can lead to inaccurately measured positions, sizes, and shapes of small targets like galaxies.

The central importance of PSF characterization to astrophysics means that there is a wealth of PSF fitters available. These generally fall into two classes: forward-modeling approaches, which use physical optics propagation based on models of optical elements, and empirical approaches, which use observed stars as fixed points to model and interpolate the PSF across the rest of the image. In both cases, the PSF model may be validated by comparing a set of reserved stars to the PSF model's prediction.

Empirical characterization tools like PSFEx (Bertin, 2011) and PIFF (Jarvis et al., 2020) [8] are widely popular in astrophysics. However, the quality of PIFF and PSFEx models tends to be quite sensitive to the values of hyperparameters used to run the software, with optimal parameter selection sometimes relying on brute-force guess-and-check runs. PIFF, using the modelling and interpolation scheme used for the Dark Energy Survey Year 3 observations, is also notably inefficient for large, well-sampled images, taking hours in the worst cases.

Because space telescopes are unimpeded by the atmosphere, their PSFs are also often characterized with forward-modeling approaches like Tiny Tim (Krist et al., 2011) and WebbPSF (Perrin et al., 2012; Perrin

[8] https://github.com/astromatic/psfex and https://github.com/rmjarvis/Piff

et al., 2014a; Ji et al., 2023). WebbPSF models are continually updated based on telescope telemetry, ensuring high accuracy in all bandpasses and for all instruments regardless of image noise. While robust, forward modeling is not infallible and may occasionally miss short-timescale variations and other "unknown unknowns" that can be captured with empirical PSF models, albeit at the cost of much higher noise.

The James Webb Space Telescope (JWST) represents a giant leap forward in our ability to explore the cosmos. Equipped with groundbreaking instruments like the Near Infrared Camera (NIRCam) and the Mid-Infrared Imager (MIRI), the telescope is poised to unlock unprecedented insights into the early universe (Robertson, 2022). At the same time, these advances usher in a new set of complexities for PSF characterization:

1. Modeling the PSFs of space observatories like JWST is non-trivial since they generally operate near their diffraction limits and exhibit intricate optical patterns that defy the analytic approximations acceptable for PSFs of ground-based imaging. In particular, the PSFs exhibit steep spatial gradients, making them highly sensitive to the hyperparameters of commonly used analytic profiles, where slight variations can lead to substantial inaccuracies. The assessment of PSF models for space-like imaging presents an additional challenge, as most metrics of fit quality, e.g., full width at half maximum or second moments of intensity, treat the PSF as an elliptical Gaussian – a poor approximation for the "spiky" NIRCam and MIRI PSFs. While there are some useful moment based metrics for assessing fit quality (e.g. Zhang et al., 2023), there are no universally adopted non-parametric diagnostics for assessing pixel-level model biases.

2. The NIRCam detector pixel scales are $0.031''$/pixel and $0.063''$/pixel for the short and long wavelength channels, respectively (Rieke et al., 2003; 2005; Beichman et al., 2012). At these fine scales, fully capturing the intensity profile of the NIRCam PSF requires a much larger number of pixels than is needed for typical surveys, whose detectors have pixel scales 3–10 times larger than NIRCam (Jarvis et al., 2020; Fu et al., 2022; York et al., 2000; McCleary et al., 2023b). As a consequence, the number of pixels need to model the full size of the PSF is much greater. This new regime prompts a necessary evaluation of how current PSF fitters perform at this new scale. That is, can the PSF fitters still capture the full dynamic range of the distortion and do so in a reasonable time.

To meet these challenges, we introduce `ShOpt`[9], a new PSF modeling tool that strives to retain the best of existing PSF modeling software while advancing their mathematical formulation and increasing their computational efficiency. Written in the high-level Julia language using a functional programming style[10], `ShOpt` offers both accessibility and speed, positioning it as a valuable tool for the astrophysical community. `ShOpt` introduces manifold-based algorithms for enhanced efficiency in analytic profile fitting. `ShOpt` also employs three distinct techniques for pixel-basis fitting: principal component analysis (PCA), an autoencoder, and kernel smoothing.

Along with `ShOpt`, we introduce a suite of non-parametric PSF fit statistics appropriate for space-based imaging, namely reduced $\chi^2$, mean relative error, and mean absolute error. Our proposed PSF characterization statistics are inspired by strong gravitational lensing analyses and move beyond the conventional metrics based on elliptical Gaussians. In this study, we evaluate the performance of `ShOpt`, PSFEx, and PIFF using data from the COSMOS-Web survey (Casey et al., 2023a), using our proposed PSF fit statistics to gauge their relative performance. In addition, we time the PSF fitters to measure the computational efficiency of each tool.

To summarize, our contributions are the following:

- We introduce `ShOpt`, and with it, a number of methods for efficient empirical PSF characterization.
- We benchmark the model fidelity of different PSF fitters using non-parametric approaches. We use real and simulated catalogs from the COSMOS-Web Survey and measure $\chi^2$, mean relative error,

[9]The name ShOpt is a contraction of "shear optimization." That is, we are finding the best-matching PSF by formulating optimization problems over the space of all possible shears. The name was inspired by the `manopt` library, a contraction of manifold optimization. It is an apt comparison given the manifold learning we describe in Section 25.1.

[10]By functional, we mean the functional programming paradigm. While Julia provides support for object-oriented design patterns with `structs`, Julia code is generally written with design patterns that put the emphasis on reusable functions.

and mean absolute error. We supplement these statistics with conventional second moment HSM fits (Mandelbaum et al., 2005; Hirata & Seljak, 2003).

- We benchmark the computational efficiency of different PSF fitters.

The remainder of this paper is structured as follows. In Section 24, we establish the workflow of `ShOpt` and the notation used in this paper. In Section 25, we develop our methods for Gaussian and pixel-basis PSF fits, and in Section 26 our methods for fitting their variation across the field of view. In Section 27, we describe our benchmarking methods and data sets. Our algorithmic choices for `ShOpt` are justified using big-$\mathcal{O}$ time complexity analysis in Section 28, specifically detailing `ShOpt`'s speed variation with different input parameters. In Section 29, we describe our benchmarking methods and data sets. We present our results in Section 30, and discussion and conclusions in Section 31.

## 24 ShOpt Notation, Workflow, and Overview

### 24.1 Notation and Preliminaries

We represent star locations in pixel coordinates as $(x, y)$ and in celestial coordinates (expressed in degrees) as $(u, v)$. In terms of a nominal position source, positive $u$ is west, and positive $v$ is to the north. "Vignette" refers to small, localized images, centered around individual stars or celestial objects, that are extracted from larger astronomical images.

We use the following notation for working with manifolds. For two sets $A$, $B$,

$$B_2 \equiv \{[x, y] : x^2 + y^2 < 1\} \subset \mathbb{R}^2 \tag{86}$$

$$\mathbb{R}_+ \equiv \{x : x > 0\} \subset \mathbb{R} \tag{87}$$

$$A \times B \equiv \{(a, b) : a \in A, b \in B\} \tag{88}$$

Throughout this paper, we describe the shape of a smoothly varying (analytic) PSF profile in terms of the variables $[s, g_1, g_2]$, where $(g_1, g_2)$ are the polarization states of an elliptical source with reduced shear $\mathbf{g} = g_1 + ig_2 = ge^{i2\theta}$, where $\theta$ is the angle of the major axis from [West/some fiducial orientation] and

$$g \equiv \frac{1-q}{1+q}, \tag{89}$$

for an ellipse with (major/minor) axis ratio $q$, such that $0 \leq g < 1$ (Bernstein & Jarvis, 2002). While shear is typically treated as a number in the complex plane $\mathbb{C}$, we make an equivalent characterization that shear is a vector $[g_1, g_2] \in \mathbb{R}^2$. This vector representation corresponds to the real and imaginary parts of the complex shear. The free parameter $s$ represents the size of the ellipse (the geometric mean of the major and minor axis lengths). We introduce free parameters $[\sigma, e_1, e_2]$ that reparameterize $[s, g_1, g_2]$, noting that we adopt a slightly different relationship between ellipticity $e$ and shear $g$ than may be seen elsewhere in the literature, dropping of a factor of 2 in Equation 99 for the purposes of an easier calculation of the inverse map (details in Section 25).

Additionally, we are often concerned with a more realistic pixel basis, where each pixel in an $n \times n$ star image is a basis element.

For our non-parametric summary statistics defined in Equations 90–93, we use $v_{i,j,k}$ to represent the $i \times j$ pixels in vignette $k$, and $p_{i,j,k}$ to represent the $i \times j$ pixels in PSF model $k$. $\sigma_{i,j,k}$ represents the uncertainty in the model at pixel $(i, j)$ and vignette $k$. The total number of vignettes is $K$. The number of pixels in a

vignette is denoted $N_{\text{pix}}$.

$$\overline{\chi^2}_{\nu} = \frac{1}{K}\sum_{k}\frac{1}{d.o.f.}\sum_{(i,j)}^{N_{\text{pix}}}\frac{(v_{i,j,k}-p_{i,j,k})^2}{(\sigma_{i,j,k})^2} \tag{90}$$

$$\text{median } \chi^2_{\nu} = \text{median }_{K}\left[\frac{1}{d.o.f.}\sum_{(i,j)}^{N_{\text{pix}}}\frac{(v_{i,j,k}-p_{i,j,k})^2}{(\sigma_{i,j,k})^2}\right] \tag{91}$$

$$\text{Mean Relative Error} = \frac{1}{N_{\text{pix}}}\sum_{(i,j)}^{N_{\text{pix}}}\frac{1}{K}\sum_{k}\frac{v_{i,j,k}-p_{i,j,k}}{v_{i,j,k}} \tag{92}$$

$$\text{Mean Absolute Error} = \frac{1}{N_{\text{pix}}}\sum_{(i,j)}^{N_{\text{pix}}}\frac{1}{K}\sum_{k}\frac{\mid v_{i,j,k}-p_{i,j,k}\mid}{\mid v_{i,j,k}\mid} \tag{93}$$

We shall henceforth refer to mean relative error as MRE and mean absolute error as MAE. To enable flux-based comparisons between PSF vignettes, which most fitters normalize to unity, and star vignettes, we add the appropriate star flux to the PSF vignettes, then add Gaussian noise with mean and variance taken from the sky background in the vicinity of the star.

### 24.2 Code Overview

`ShOpt` takes inspiration from robotics algorithms, such as SE-Sync (Rosen et al., 2019a), that run on manifold-valued data. The manifold properties of shears are described in Bernstein & Jarvis (2002); we expand on their work to provide more robust multivariate analytic fits to PSF intensity profiles. We specifically use multivariate Gaussians, as they are cheap to compute, but for a more rigorous treatment of optimization methods on manifold-valued data, see Absil et al. (2008a) and Boumal (2023). `ShOpt` also provides three modes for fitting PSFs in the pixel basis: `PCA` mode, `autoencoder` mode, and `smoothing` mode, detailed in Section 25. `PCA` mode approximates the original image by summing the first $n$ principal components, where $n$ is supplied by the user. We also introduce `autoencoder` mode, which uses a neural network with an autoencoder architecture to learn the PSF. A square image of side length $n$ can be thought of as vectors in $\mathbb{R}^{n^2}$, where each pixel is a basis element. The architecture is built so that the image is encoded into a vector space represented by a basis with $\dim(V) < n^2$ before being decoded back into the dimension of our original image. The nonlinearity of the network ensures that the key features are learned instead of some linear combination of the sky background. Both `PCA` and `autoencoder` modes provide the end user with tunable parameters that allow for accurate reconstruction of the model vignettes without overfitting to noise. `Smoothing` mode applies a Lanczos kernel to the input vignettes, and uses the smoothed output for the pixel basis fit.

**Why Julia?**

`ShOpt` is written in Julia. The Julia programming language is a high-level and functional language like Python, which makes it accessible to a community of open-source developers. At the same time, Julia is equipped with a just-in-time compiler, which helps Julia code execute quickly by recycling compiled code. This offers a speed advantage over Python, which is first transpiled to C before being translated into machine code. Julia also offers some of the most sophisticated tools for problems at the intersection of numerical linear algebra and optimization, such as the `ForwardDiff.jl` and `Optim.jl` (Revels et al., 2016; Mogensen & Riseth, 2018) libraries. Julia also has an abundance of support for working with manifolds such as `manopt`, which may be pertinent in future releases of `ShOpt` (Bergmann, 2022). Finally, much of today's production code is written with the help of program synthesis tools such as GitHub Copilot. It is clear that these tools will soon make Julia more accessible than ever before to astrophysicists and other potential future `ShOpt` contributors. Work is being done to strengthen these

tools for programming languages with much less training data available, such as `Julia`. Cassano et al. (2023) successfully demonstrated how to transfer knowledge from programming languages with lots of publicly available training data (e.g., Python) to programming languages with much less training data available (e.g., Racket, OCaml, Lua, R, and most importantly, Julia).

The source code for `ShOpt` can be found on GitHub[11] and is detailed in our companion paper (Berman & McCleary, 2024b).

`ShOpt` accepts as input a FITS-format catalog containing star vignettes, positions, and signal-to-noise ratios (SNR). One concession `ShOpt` makes is that it makes more sense to build on the `Astropy` infrastructure than to rebuild everything from scratch. As such, `PyCall.jl` is used to handle FITS input and output. Since this is only done once and not iterated over in any main loop, this does not significantly affect performance.

After an initial quality check based on SNR, the star vignettes are fit with a multivariate Gaussian to remove outlier stars. While these Gaussian fits are not used to create a PSF model, they are helpful in screening out stars that are saturated, too bright, or too faint. In situations where the image noise is high, the SNR threshold must be kept low to include enough stars for fitting. After these pre-processing steps, stars are fit in the pixel basis using either the `PCA` or `autoencoder` modes. The output learned stars are then fit with a multivariate Gaussian, which serves both to reject bad pixel-basis fits and to record the rough size and shape of the PSF. Surviving stars act as fixed points for polynomial interpolation of the PSF's spatial variation across the field of view. Finally, the learned data is saved to a summary FITS file, along with diagnostic plots and logging statements. This is summarized in Algorithm 4.

**Algorithm 4** ShOpt Workflow

```
starCatalog ← loadStarCatalog()
filteredCatalog ← filterBySignalToNoise(starCatalog)
for each vignette in filteredCatalog do
    gaussianFit ← fitMultivariateGaussian(vignette)
    if not isGoodFit(gaussianFit) then
        remove(vignette)
    end if
end for
pixelGrid ← createPixelGrid(filteredCatalog)
reducedData ← dimensionalityReduction(pixelGrid)
for each grid in reducedData do
    gaussianFit ← fitMultivariateGaussian(grid)
    if not isGoodFit(gaussianFit) then
        remove(grid)
    end if
end for
for num iterations do
    for each point in fieldOfView do
        interpolatedStar ← interpolate(point, reducedData)
        if isOutlier(interpolatedStar) then
            remove(Star)
        end if
    end for
end for
plotAndSaveData()
=0
```

[11] https://github.com/EdwardBerman/shopt and https://edwardberman.github.io/shopt/

# 25 Parameter Estimation for PSF Modeling

## 25.1 Analytic Profile Fitting

`ShOpt`'s analytic model has two components, a shear transformation and a radial function. We elect to fit an elliptical Gaussian $f_{Gaussian}(r)$ for our radial function:

$$f_{Gaussian}(r) = Ae^{-r^2/2} \tag{94}$$

where A makes the image sum to unity. There are other radial profiles that we could choose from, however, any radial profile parameterized by $[s, g_1, g_2]$ contains some azimuthal symmetry that makes the "wings" of the PSF difficult to model. Thus, it is not worth the computational cost to fix a radial profile that is any more elaborate than an elliptical Gaussian as there are diminishing returns for accuracy.

For any analytic model, the shear is always the same,

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \frac{s}{\sqrt{1-(g_1)^2-(g_2)^2}} \begin{bmatrix} 1+g_1 & g_2 \\ g_2 & 1-g_1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{95}$$

$$\tag{96}$$

and

$$r = \sqrt{(u')^2 + (v')^2}. \tag{97}$$

The shear matrix in Equation 95 is known to be positive definite, and $s$ is strictly positive. Therefore, our parameters are constrained to $\mathbb{R}_+ \times B_2$. The parameter space can be visualized as a solid cylinder extending infinitely far from the origin in one direction, as seen in Figure 28.

The finite domain of $g$ poses a problem for fitting. As such, we adopt a fitting parameter $e$ that allows us to map any value in $\mathbb{R}^2$ onto $g$:

$$e_i = \frac{g_i}{1-||g||^2}. \tag{98}$$

This map[12] from $B_2 \to \mathbb{R}^2$ is relatively smooth, so that any automatic differentiation software can handle it during the fitting process. We can construct an inverse map via

$$||g||^2 = \frac{2||e||^2}{1+2||e||^2+\sqrt{1+4||e||^2}}. \tag{99}$$

The inverse map allows us to constrain our update steps inside of our parameter space which leads to quicker convergence and the ability to handle noisier data more effectively. Equation 99 also keeps the nice properties of smoothness and monotonicity desirable for activation functions, see Figure 29. Equation 99 is not the canonical map from $\mathbb{R}^2$ to $B_2(r)$; two sigmoid functions could also have been used. We chose this particular function because it avoids the finite domain problem, is easily differentiable by the tools we use, and has a loose geometric interpretation described in (Bonnet & Mellier, 1995).

Note that this inverse only specifies what the new norm should be; the components still need to be adjusted accordingly. We reparameterize[13] as follows:

$$s \equiv \sigma^2 \tag{100}$$

$$g_i \equiv e_i \frac{||g||}{||e||} \tag{101}$$

The values of $\sigma, e_1, e_2$ are obtained from each update step and $\|e\|$ is then determined from the usual $L_2$ norm. The parameter $s$ is obtained from Equation 100 and $\|g\|$ from Equation 99. Finally, $\|g\|$ is used with $\|e\|$ to calculate $g_1$ and $g_2$.

[12] We could look at this as a map from $B_n \to \mathbb{R}^n$ but in this case we are only concerned about vectors $g = [g_1, g_2]$ and $e = [e_1, e_2]$

[13] Technically, for $s$ to be strictly positive we would set $s := \sigma^2 + \varepsilon$, where $\varepsilon$ is some small positive perturbation. Numerically, it makes little difference: stars with very small $s$ are removed during pre-processing.

We have that $r$ is a function of $[u, v, s, g_1, g_2]$ and $f$ is a function of $r$ such that

$$f(r) \equiv I_p(u, v, s, g_1, g_2). \tag{102}$$

Even though we solve for parameters $[\sigma, e_1, e_2]$, the loss is still dictated by $[s, g_1, g_2]$.

$$\text{cost}(s_t, g_{1_t}, g_{2_t}) = \frac{1}{2} \sum_{u_{pix}} \sum_{v_{pix}} (I_p(u, v)^* - I_p(u, v, s_t, g_{1_t}, g_{2_t}))^2 \tag{103}$$

Here $*$ denotes the ground truth (i.e. star) in the star vignette and the subscript $t$ denotes iteration $t$ in an LBFGS run. Parameters are found using Julia's `Optim.jl` library (Mogensen & Riseth, 2018) and the gradient is computed using Julia's `ForwardDiff.jl` library (Revels et al., 2016).

### 25.2 Pixel Grid Fits

`PCA` mode, `autoencoder` mode, and `smoothing` mode all provide "pixel-grid" mechanisms for reconstructing the PSF in the image pixel basis. We outline each of these modes below.

#### 25.2.1 PCA Mode

`PCA` mode approximates a star image with a rank-$n$ approximation of the original image using principal component analysis, where $n$ is supplied by the user. Modeling the vignette in this way gives the user a method of determining how much they want their pixel grid model to represent the original star vignette. Choosing a lower rank approximation will yield less detail on the shape of the PSF, but will do so quickly and without much noise contamination. On the other hand, a higher rank approximation will capture more of the fine details, but at the risk of capturing some of the noise. It should be noted that the choice of $n$ has little overhead cost on the `ShOpt` computation time.

To minimize aliasing effects that might appear in low-rank approximations, the output image is convolved with a smoothing Lanczos kernel $\mathcal{L}$:

$$\mathcal{L}(x, a) = \begin{cases} 0 & \text{if } |x| \geq a \\ 1 & \text{if } x = 0 \\ a\text{sinc}(\pi x)\text{sinc}\left(\frac{\pi x}{a}\right) & \text{otherwise} \end{cases}, \tag{104}$$

where $a$ is a tunable parameter for the size of the kernel and $x$ is the distance from the center of the kernel.

#### 25.2.2 Autoencoder Mode

`Autoencoder` mode uses deep learning to reconstruct PSF vignettes pixel by pixel. We adopt an autoencoder architecture because the projection into a latent space forces the machine learning algorithm to learn the key features of the image regardless of the noise present. The operations are carried out by Julia's `Flux.jl` machine learning library (Innes, 2018b).

In our specific architecture, the input star vignette is first flattened and then passed through the network, which has an encoder with one layer containing 128 nodes and a second layer containing 64 nodes. The encoder feeds into a latent space with 32 nodes, which is then decoded back into a layer of 64 nodes, 128 nodes, and finally back into an image (vignette) of the same dimension as the input vignette. We enforce both the input and output vignettes to sum to unity so that the relative distributions of intensities can be compared. The loss function is a mean squared error:

$$\text{cost}(x) = \frac{1}{N_{\text{pix}}} \sum_{N_{\text{pix}}} (x_p - \hat{x}_p)^2, \tag{105}$$

where $\hat{x}$ denotes the image after it has been put through the autoencoder.

The network consists of two activation functions. A `leakyrelu` (Maas, 2013) function is used for all layers except for the last one. This choice reflects that most of the vignette pixel values are positive, and the

ones that are not are usually close to zero or eliminated in data pre-processing. The final layer uses a `tanh` activation function to ensure that output values stay bounded between $(-1, 1)$. The network trains until it either hits a specified number of epochs or until it hits a stopping gradient. We encourage exploration of the number of epochs and the minimum stopping gradient to find an appropriate middle ground between accuracy and time of completion. Stopping gradients between $10^{-5}$ and $10^{-6}$ are usually sufficient to get a good approximation in reasonable time and in 100 epochs or less. This was revealed through the use of diagnostic plots that are introduced in Section 30. The activation functions and number of nodes are not tunable by default; changing those is not recommended.

### 25.2.3 Smoothing Mode

In the case of well-sampled images like the NIRCam data considered in this analysis, we find that a basic smoothing is sufficient before interpolation. This is implemented in ShOpt as `Smoothing` mode, which uses only the Lanczos kernel introduced in Equation 104 before produce the pixel grid fit. While `PCA` and `Autoencoder` can yield denoised PSF models from low dimensional reconstructions, `Smoothing` mode is a simpler technique that avoids some of their limitations, albeit at the expense of noisier models.

## 26 Interpolation Across the Full Field of View

To fit for the spatial variation of the size and shape of the PSF, `ShOpt` first produces a rough estimate by interpolating the analytic fit parameters $[s, g_1, g_2]$ across the field of view using a polynomial of order $n$, where $n$ is supplied by the user. For an order-$n$ polynomial $p(u, v)$), the cost function takes the form

$$\text{cost}_p(\left[a_0, \ldots, a_{(n+1)(n+2)//2}\right]) = \left( \left[ \sum_{(i,j), i+j \leq n} a_{ij} u^i v^j \right] - p(u, v)^* \right)^2 \tag{106}$$

where $*$ denotes the ground truth obtained by pixel basis fits. This gives us different polynomials in $u$ and $v$ for $[s, g_1, g_2]$. There is a tunable parameter for the stopping gradient that leaves the tradeoff between speed and accuracy of the polynomial interpolation to the end user. As elsewhere, the minimum of the cost function is found with LBFGS and the `Optim.jl` library.

Subsequently, the pixel-grid model is interpolated across the field of view. By definition, each pixel in pixel-grid models is a basis element, so the natural thing to do is to give each pixel in an $n \times n$ vignette its own polynomial. However, we found that this approach for solving for the polynomial coefficients produced systematically biased models, possibly due to the conditioning of the pixel intensity values. Thus, we opt for an alternative approach. To solve for the coefficients of the polynomial in the pixel basis fit, we may specify an matrix $m \times n$ matrix, where $m$ is the number of stars we are interpolating over and $n$ is still order of the polynomial. We denote this design matrix by $A$. If the vector $x$ represents the coefficients of the polynomials and the vector $b$ represents the intensity values, then we may use the known least squares solution to the matrix equation

$$Ax = b. \tag{107}$$

Not all stars are good exemplars of the PSF due to things like saturation, color effects, and noise. To combat this, `ShOpt` does its polynomial interpolation over several iterations according to Algorithm 5.

**Algorithm 5** Iterative Polynomial Interpolation and Star Filtering Process

NumIterations ← [define number of iterations]
**for** $i \leftarrow 1$ **to** NumIterations **do**
    PSFModel ← PolynomialInterpolate(TrainingStars)
    RenderedStars ← RenderPSF(PSFModel)
    MSE ← ComputeMSE(TrainingStars, RenderedStars) {MSE: Mean Squared Error}
    WorstStars ← GetWorstPerformingStars(MSE, 10% or N sigma)
    TrainingStars ← TrainingStars - WorstStars
**end for**
FinalPSFModel ← PolynomialInterpolate(TrainingStars)
**return** TrainingStars, FinalPSFModel =0

The number of iterations to refine the polynomial interpolation is specified by the user. After each iteration, the predicted PSF is rendered at each star location and the mean squared error (Equation 108) is computed using the training stars. The worst 10% of training stars are filtered out.

$$\text{Mean Squared Error} = \frac{1}{N_{\text{pix}}} \sum_{(i,j)}^{N_{\text{pix}}} (v_{i,j} - p_{i,j})^2 \tag{108}$$

Alternatively, we provide a mode that removes $N \times \sigma$ outliers from the interpolation. Polynomial interpolation for high-degree polynomials can be the most expensive part of the whole PSF fitting procedure. For this reason, `ShOpt` is strict about filtering outliers in polynomial interpretation. Training stars are excluded based on the value of $s$ that was found during the analytic-profile interpolation step, which eliminates the need for additional iterations to clean the training data. Additionally, we make the conscious decision not to continue filtering our training stars until we reach a given $n\sigma$ confidence. The JWST PSFs contain lots of background noise between the wings of the stars we are trying to model. Consequently, an excessive number of iterations may be required to reach the desired level of confidence.

## 27 Data pre-processing and outlier rejection

In `ShOpt`'s outlier rejection process, three distinct phases are employed: (1) SNR-based filtering on input stars; (2) a Gaussian fit size filter on the input stars; and (3) a separate Gaussian fit size filter on pixel grid models. Following these phases, the process advances to the iterative refinement for polynomial fitting, as detailed in Section 26.

Before doing anything else, `ShOpt` filters out bad stars on the basis of SNR; the default SNR metric is the SExtractor `SNR_WIN` parameter. By default, `ShOpt` filters out the noisiest 33% of entries, a percentage determined through experimentation. Users can adjust this threshold to fit their specific data sets.

The remaining vignettes are then fit with a multivariate Gaussian, as described in Section 25. We filter again based on the object size $s$. A very small or very large $s$ probably corresponds to objects that are not point sources. By default, vignettes with $s < 0.075$ or $s > 1$ are filtered out. This ensures further computation time is not wasted on bad data. We apply the same size filtering after fitting Gaussians to the pixel grid models. This prevents poor pixel grid models from being used in our polynomial interpolation, as mentioned in Section 26.

`ShOpt` also offers several methods for efficiently cleaning useful data without discarding it. Given that SExtractor sets the flux of interloping sources to a sentinel value of $-10^{32}$, `ShOpt` sets pixels less than $-1000$ to `NaN`. Before doing any analytic or pixel grid fits, we also smooth the image according to the kernel introduced in Equation 104 to avoid any hot pixels in the vignettes. Finally, we recommend that users select `true` in the yaml configuration for the setting `sum_pixel_grid_and_inputs_to_unity`. This enforces intensity to sum to unity in each of our model vignettes and their pixel grid fits.

# 28 Runtime Analysis

In this section, we compare the algorithmic complexity and convergence properties of `ShOpt`, PIFF and PSFEx, examining the analytic profile fits, pixel grid fits, and polynomial interpolation steps separately. Runtime analysis of the three PSF fitters allow us to argue for the algorithmic choices implemented in `ShOpt` without needing to factor in programming language or computing power. The results will serve as predictions for the speed tests of Section 30.3.

## 28.1 Analytic Profile Fit Runtime

Let $n$ denote the number of pixels on one side of a square vignette, so that there are $n^2$ total pixels. In our optimization scheme, we compute the loss between every pixel in the vignette and the analytic profile prediction an average of $I$ times. Our nominal runtime is thus $\mathcal{O}(n^2 \times I)$.

The PIFF configuration used in this work does not solve for $[s, g_1, g_2]$ using a nonlinear process. Instead, this configuration uses iterative linear least squares with slight updates to the centroid and total flux over a grid of pixels. However, there are other PIFF configurations that use nonlinear processes to find $[s, g_1, g_2]$ based off first order approximation methods. On the other hand, `ShOpt` uses LBFGS, which uses super-linear approximations to calculate the direction of improvement to find $[s, g_1, g_2]$. The LBFGS algorithm and the reparameterization step introduced in Section 25.1 allows us to argue that $I_{ShOpt} < I_{PIFF}$. While we refrain from claiming that `ShOpt` will always converge to the correct $[s, g_1, g_2]$ faster than PIFF, we have designed `ShOpt` with the hopes that using a more memory expensive technique to compute its update steps while keeping the solution within a constraint gives us better convergence.

## 28.2 Pixel Grid Fit Runtime

### 28.2.1 PCA Runtime

Principal component analysis relies on computing the singular value decomposition of the covariance matrix of a given data set. Singular value decomposition is $\mathcal{O}(n^3)$ for an $n \times n$ matrix, but the fit is typically much cheaper to compute for an approximation using the first $k$ principal components. While this does not lower the big-O complexity to the $\chi^2$ minimization pixel-grid fitting implemented in PIFF and PSFEx (Bertin, 2011; Jarvis et al., 2020), we do not encounter any noticeable speed bottle necks at this step.[14] We will explore this more in in 30.3.

### 28.2.2 Autoencoder Runtime

As in 28.1, we may observe that the complexity is $\mathcal{O}(n^2 \times I)$ because the loss is computed an average of $I$ times over $n^2$ pixels. The number of parameters between layers as a function of neurons $m$ in layer $i$ in our network is given by

$$N_{params} = (m_{i-1} \times m_i) + m_i \tag{109}$$

where the first term corresponds to the number of weights and the second term corresponds to the biases. In our network the number of nodes for the input and output is set by a flattened version of the input image. Therefore, the number of parameters to learn grows as $\big[\big(n^2 \times 128\big) + 128\big] + \big[\big(128 \times n^2\big) + n^2\big]$. On the other hand, PIFF and PSFEx employ a form of $\chi^2$ minimization, wherein each pixel in the image is a learnable parameter; for an $(n, n)$ image there are only $n^2$ learnable parameters and $n^2$ pixels computed in the loss function. Even though the loss function in `ShOpt` is also computed over $n^2$ pixels, there are much more than $n^2$ learnable parameters, so we expect the autoencoder to take more iterations to converge on average than PIFF and PSFEx. For this reason, `autoencoder` is not the default mode for pixel grid fits in `shopt`. It should be reserved for cases that demand precision and where the images are small enough to be learned efficiently so the transfer learning effect is more pronounced.

[14] A $\chi^2$ minimization pixel-grid fit will be incorporated in a future `ShOpt` release.

### 28.3 Polynomial Interpolation Runtime

Almost all PSF fitting software makes use of polynomial interpolation to model the variation of the PSF across the camera's field of view. Any performance gains in this area are primarily derived from the efficiency of the polynomial fitting process to the data. In PSFEx, these fits are implemented using PCA, which Bertin (2011) argues requires the lowest number of basis vectors to approximate an image if the data is sufficiently well-behaved.

In `ShOpt`, we use the design matrix and the known least squares solution to the matrix equation $Ax = b$ to fit each pixel with an independent polynomial. This process is inherently parallelizable because the polynomial found for one pixel is independent of the polynomial found for any other pixel. `ShOpt` will automatically use as many threads are made available to it. The idea to break these operations into computation blocks is inspired by Tiny Machine Learning problems described in Sabot et al. (2023) and the CAKE algorithm outlined in Kung et al. (2021). Currently, the number of threads is not configurable because it has to be specified before the program is run. On UNIX, we ran `export JULIA_NUM_THREADS=auto` before running the program, on Windows you can similarly run `set JULIA_NUM_THREADS=auto`.

### 28.4 Order of Operations

We are also deliberate about the order of operations in `ShOpt`. The initial analytic profile fitting serves as a data cleaning method, which allows us to not waste compute time fitting pixel-grid models to outlier stars. A second round of analytic profile fits to the resulting PSF models further refines the data set, ensuring that the (computationally expensive) polynomial interpolation is applied only to the highest quality PSF models. By contrast, PIFF adopts an integrated approach where analytic and pixel grid fits are interwoven with iterations of polynomial interpolation. While this method is thorough, our approach is designed to prioritize computational expediency.

## 29 Benchmarking, Data, and Analysis

Our PSF analysis utilizes NIRCam imaging from the COSMOS-Web survey (Casey et al., 2023a), which we have chosen for its expansive coverage (more than three times the area of all other JWST surveys combined) in four bandpasses as well as the fact that many of its science cases require careful PSF characterization. This section presents an overview of the three COSMOS-Web data sets employed in our PSF benchmarking analysis—simulated single exposures, simulated observation mosaics, and real observation mosaics—followed by a description of the benchmarking methods we apply to each.

We employ simulated observations for PSF code unit testing because they provide a controlled environment with fully known input parameters. Unlike real observations, simulations allow for us to control for observational imperfections such as saturation, noise, or defective pixels; this allows for the establishment of a ground truth against which the accuracy of PSF model fits can be measured directly. Although real observations feature the true PSF and are integral to our study, their inherent uncertainties and the absence of a definitive PSF ground truth make them less suitable for initial code validation. Instead, the real data serves as a stress-test for the PSF fitting codes, ensuring they remain effective when confronted with the vagaries of real observations.

1. **Simulated Data:** These simulations are based on the COSMOS 2020 data (Weaver et al., 2022), and contain all the known galaxies and stars in this field. Galaxy fluxes in JWST filters were calculated from `Stardust` SEDs if they existed (Kokorev et al., 2021); otherwise, fluxes were found by interpolating across existing photometry in other filters. Star fluxes were similarly modeled by interpolating available photometry. Galaxies with counterparts in the Zurich Structure & Morphology catalog (Scarlata et al., 2007; Sargent et al., 2007) were assigned the according morphological parameters (Sérsic indices, ellipticity, position angle, size), or from observed distributions if they did not.

   COSMOS-Web-like NIRCam images were generated using the Multi-Instrument Ramp Generator (MIRAGE; Hilbert et al., 2019), which includes PSF modeling with WebbPSF (**?**), sky background,

detector noise, dark current, and Poisson noise. After the images were generated, the JWST calibration pipeline (Bushouse et al., 2023) was used to reduce the raw NIRCam data and create mosaics, with some modifications, like 1/f noise and subtraction of low-level background (e.g., Finkelstein et al., 2022; Bagley et al., 2022). We employ two sets of simulated images:

(a) **Simulated Single Exposures:** Stars are generated by convolving a MIRAGE WebbPSF model with an idealized point sources, so these images allow us to measure success directly by comparing the learned PSF to the input PSF. We employ images from stage 2 of the JWST calibration pipeline, so there are no effects from dithering or distortion (i.e., `tweakreg`). The main challenge in using this data is low star density, particularly for the short wavelength channel. With a training/validation split of 90%, this led to an average of 1 validation star for each subarray in the F115W filter, 1 validation star for the F150W filter, 2.2 stars for the F277W filter, and 2.3 stars for the F444W filter. To gather enough data for meaningful summary statistics, we combine exposures from 156 different visits.

(b) **Simulated Mosaics:** Image mosaics are `i2d`-format data cubes built from single exposures that have passed through stage 3 of the JWST science calibration pipeline.
As such, the mosaics do reflect normal dithering and distortion effects. The simulated mosaics used in our study cover a contiguous area of 76 arcmin$^2$ at the full COSMOS-Web depth ($\sim$ 27th magnitude) and provide a reasonable 0.5 star per square arcminute.

For an in-depth discussion of the simulation process, refer to Drakos et al. (in prep).

2. **Real Mosaics:** The real data used in our analysis was taken in April 2023 and includes visits $77-152$, covering 0.28 deg$^2$ in the bottom right of our allocated area of the COSMOS field (Casey et al., 2023a), JWST program ID 1727[15]. Three of the planned visits were skipped, so the data includes a total of 72 visits. As with the simulated mosaics, we use the `i2d`-format data cubes produced by stage 3 of the JWST science calibration pipeline. We restrict ourselves to an approximately 0.11 deg$^2$ area of the full field of view, corresponding to tiles $\{A1, A5, A6, A10\}$ in Figure 31; we chose these particular tiles to test the relative performance of the PSF fitters where we expect astrometric distortions to be the most severe.

Our benchmarking procedure has four steps:

1. Run Source Extractor (SExtractor) on our images to generate star catalogs (Bertin & Arnouts, 1996). For the simulated data, stars are identified by matching the SExtractor catalogs the input point source catalogs. For real data, star catalogs are created using cuts on the stellar locus, an example of which is shown in Figure 32. The config file for SExtractor is given in the Appendix. Note that we run source extractor on the individual tiles and aggregate our results over all of them.

2. The resulting star catalogs are segregated into training (90%) and validation (10%) catalogs. We are careful to filter out saturated stars in our catalogs by searching for sentinel values of 0 in the `ERR` extension (for i2d-format mosaics) or $1, 2$ in the `DQ` extension (for single exposures). Vignettes that contain any pixels set to a sentinel value are removed from the catalogs.

3. We use the training catalogs to get empirical PSF models for each fitter. We run `ShOpt` (Berman & McCleary, 2024b), PIFF (Jarvis et al., 2020), and PSFEx (Bertin, 2011).

4. We then use the reserved validation star catalogs to calculate the summary statistics of Equations 90–93. Figure 33 illustrates this process: we calculate the residuals between the star vignettes (or MIRAGE cutouts for simulations) and renderings of the PSF models, then take the mean. To obtain single residual scores, we compute the mean and standard error of all pixels in residual images.

The code for creating star catalogs, calculating statistics, and creating the associated PSF diagnostic figures can be found on GitHub[16].

[15] Available from MAST at STScI, `http://mast.stsci.edu`
[16] https://github.com/mcclearyj/cweb_psf

| Data Type | Filter | Validation Stars |
|---|---|---|
| Simulated single exposures | F115W | 3373 |
| | F150W | 7186 |
| | F277W | 2573 |
| | F444W | 740 |
| Simulated mosaics | F115W | 37 |
| | F150W | 79 |
| | F277W | 35 |
| | F444W | 35 |
| Real mosaics | F115W | 155 |
| | F150W | 156 |
| | F277W | 148 |
| | F444W | 136 |

Table 4: Number of reserved validation stars for the three imaging data sets under consideration in each of the four COSMOS-Web NIRCam bandpasses.

## 30 Results

In this section, we report the outcomes of the PSF benchmarking analysis detailed in Section 29. The comprehensive PSF model fidelity analysis is presented in Section 30.1, and the assessment of the computational efficiency of the different PSF fitters is presented in Section 30.3.

We note that `ShOpt` was run in `smoothing` mode throughout. We found that `PCA` mode and `Autoencoder` mode added additional complexity to the PSF fitting process that resulted in poorer fits. Appendix C contains representative configuration files supplied to each PSF fitter.

### 30.1 Non-parametric Model Fidelity

As described above, we evaluate the relative performance of PSF models produced by PIFF, PSFEx, and `ShOpt` using mean and median reduced $\chi^2$ residuals, the MRE, and the MAE (cf. Equations 90–93). These statistics are computed using the reserved validation stars, which number from 35 for the simulated F277W and mosaics to 156 for the F150W real mosaics (Table 4). Statistics and diagnostic plots are based on PSF vignette sizes of $(75, 75)$ pixels, which encloses the majority of the relevant star and PSF light profiles without including excessive sky background or a large number of interloping objects.

#### 30.1.1 Simulated single exposures

While the simulated single exposures yield ample training data in aggregate, training data on individual detectors from individual visits is sparse. Despite this sparsity of training data, Figures 34, 85, and 86 show that both PSFEx and `ShOpt` produce reasonable models of the PSF. While both PSF fitters seem to slightly underfit the center of the PSF, they are otherwise able to model the finer details of the PSF.

The distribution of reduced $\chi^2$ shown in Figure 35 illustrates that `ShOpt` produces a fit that is just as strong, if not stronger than PSFex.

#### 30.1.2 Simulated mosaics

The simulated data mosaics have a higher density of training stars, as evinced by significantly better fits for PSFEx and `ShOpt` than the simulated single exposures. We also supply PIFF fits for these images. Figures 36 and 87–89 do not show the same over-concentration visible in Figures 34, 85, and 86.

Figure 37 suggests minimal statistical difference in performance among the PSF fitters, with the same heavy tailed distribution of $\chi^2_\nu$ for each.

| Filter | PSF Fitter | MAE | MRE | $\overline{\chi^2_\nu}$ | Median $\chi^2_\nu$ |
|---|---|---|---|---|---|
| F115W | ShOpt | $3.41^{+5.17}_{-1.79}$ | 0.80±1.18 | $8.89^{+25.85}_{-1.16}$ | 1.78 |
| | PIFF | $3.11^{+4.78}_{-1.72}$ | 0.87±1.20 | $12.76\ ^{+49.40}_{-1.26}$ | 1.82 |
| | PSFEx | $2.67^{+3.90}_{-1.54}$ | 0.81±0.94 | $3.81^{+10.90}_{-1.26}$ | 1.88 |
| F150W | ShOpt | $3.22^{+4.39}_{-2.08}$ | 0.88±0.79 | $5.82^{+3.73}_{-0.93}$ | 1.42 |
| | PIFF | $2.87^{+3.83}_{-2.01}$ | 0.88±0.72 | $5.93^{+4.90}_{-0.94}$ | 1.39 |
| | PSFEx | $2.70^{+3.54}_{-1.90}$ | 0.87±0.64 | $5.40^{+2.07}_{-0.92}$ | 1.30 |
| F277W | ShOpt | $3.60^{+5.49}_{-1.83}$ | 0.77±1.26 | $15.21^{+20.29}_{-1.25}$ | 1.98 |
| | PIFF | $2.75^{+4.01}_{-1.68}$ | 0.85±1.06 | $36.97^{+48.55}_{-1.38}$ | 2.09 |
| | PSFEx | $2.67^{+3.92}_{-1.58}$ | 0.83±0.96 | $16.99^{+37.48}_{-1.12}$ | 1.88 |
| F444W | ShOpt | $3.08^{+4.43}_{-1.86}$ | 0.80±1.07 | $6.11^{+6.14}_{-1.03}$ | 1.70 |
| | PIFF | $2.84^{+4.10}_{-1.76}$ | 0.89±1.05 | $13.49^{+15.76}_{-1.12}$ | 1.71 |
| | PSFEx | $2.76^{+3.99}_{-1.70}$ | 0.86±1.00 | $2.01^{+3.31}_{-1.03}$ | 1.46 |

Table 5: Simulated mosaic summary statistics. The MAE and $\overline{\chi^2_\nu}$ statistics are reported with 10% and 90% errors.

Table 5 shows MRE and MAE values are consistent with zero for all `ShOpt` and PSFEx models, suggesting minimal bias. While the PIFF model residuals are also consistent with zero, the 90% bounds tend to be larger than `ShOpt` and PSFEx, indicating an increased incidence of catastrophic fits. We also point out that the 10% errors tend to be close the median error among each of the PSF fitters.

In general, the presence of outlier fits obscures a clear ranking based on MAE, MRE, and $\overline{\chi^2}$ alone. The median $\chi^2$ and its distribution, as shown in Figure 37 and Table 5, seem to be more indicative of performance, demonstrating no significant difference in the reliability of the PSF fitters for the simulated mosaics.

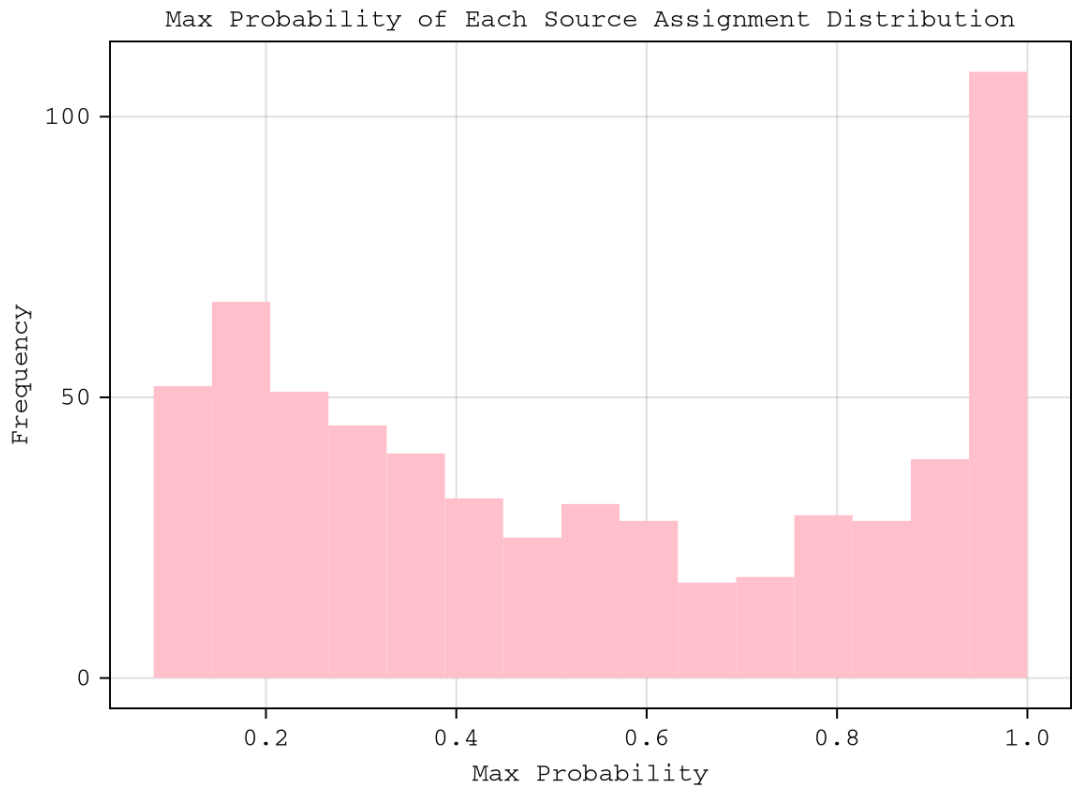
### 30.1.3 Real mosaics

We find similar results for the real data mosaics as for the simulated mosaics exposures, namely that each PSF fitter produces similarly high quality models. Figures 38 and 90-91 suggest that in the main, both `ShOpt` and PSFEx are able to model the finer details of the PSF at all bandpasses analyzed.

The high model fidelity for each fitter is further supported by the values of median and mean reduced $\chi^2$ in Table 6, as well as the distributions of reduced $\chi^2$ in Figure and 39. We do not run PIFF on the real mosaics due to timing costs; the real data mosaics cover tens of thousands of pixels, and PIFF fits did not reliably converge.

| Filter | PSF Fitter | MAE | MRE | $\overline{\chi^2}_\nu$ | Median $\chi^2_\nu$ |
|---|---|---|---|---|---|
| F115W | ShOpt | $2.96^{+3.65}_{-2.25}$ | 0.88±0.50 | $3.12^{+3.96}_{-0.99}$ | 1.49 |
| | PSFEx | $2.76^{+3.38}_{-2.15}$ | 0.90±0.49 | $4.49^{+5.30}_{-0.96}$ | 1.45 |
| F150W | ShOpt | $2.96^{+3.69}_{-2.24}$ | 0.86±0.51 | $2.05^{+3.00}_{-1.02}$ | 1.43 |
| | PSFEx | $2.76^{+3.42}_{-2.15}$ | 0.87±0.48 | $1.65^{+2.09}_{-0.93}$ | 1.27 |
| F277W | ShOpt | $2.96^{+3.92}_{-1.87}$ | 0.63±0.55 | $29.28^{+92.44}_{-2.59}$ | 4.15 |
| | PSFEx | $2.56^{+3.27}_{-1.81}$ | 0.75±0.50 | $13.55^{+34.46}_{-2.46}$ | 3.60 |
| F444W | ShOpt | $2.89^{+3.83}_{-1.91}$ | 0.70±0.60 | $10.28^{+32.84}_{-1.67}$ | 2.86 |
| | PSFEx | $2.58^{+3.30}_{-1.84}$ | 0.78±0.52 | $7.34^{+15.22}_{-1.71}$ | 2.49 |

Table 6: Real mosaic summary statistics.



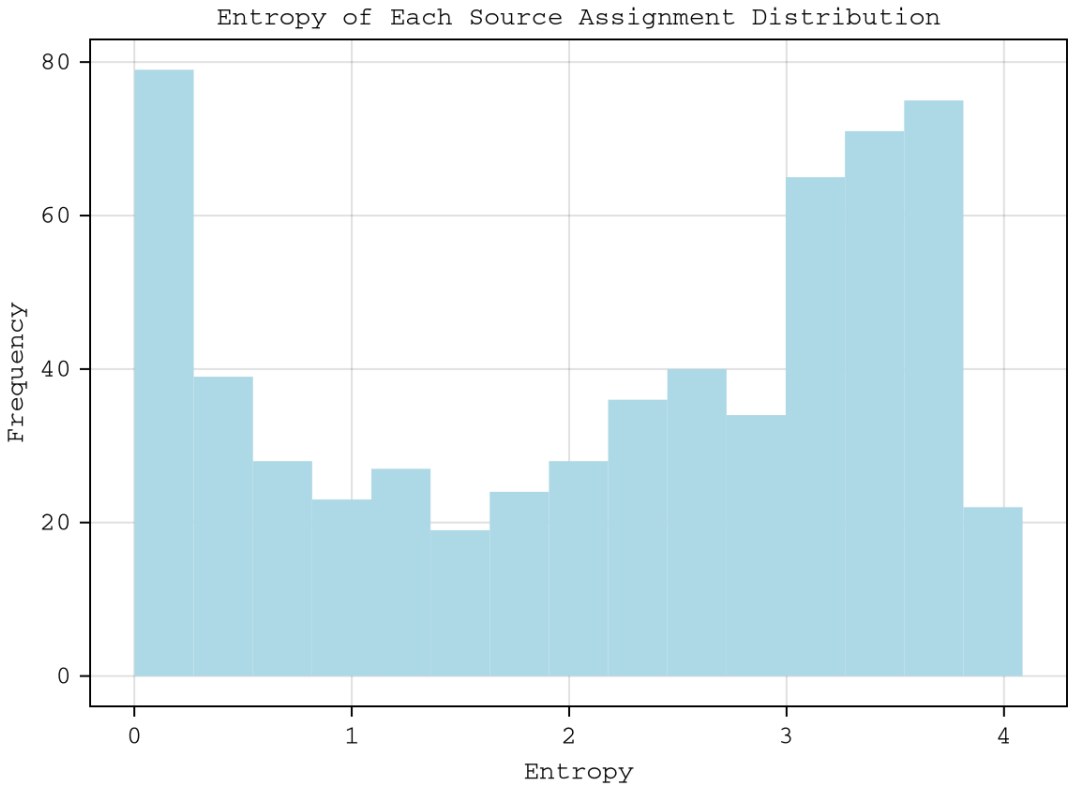




Figure 9: The maximum probability and entropy of the source assignment distributions. Each column in $U$ corresponds to a probability distribution. For each of those distributions, we compute the maximum probability and Shannon entropy.

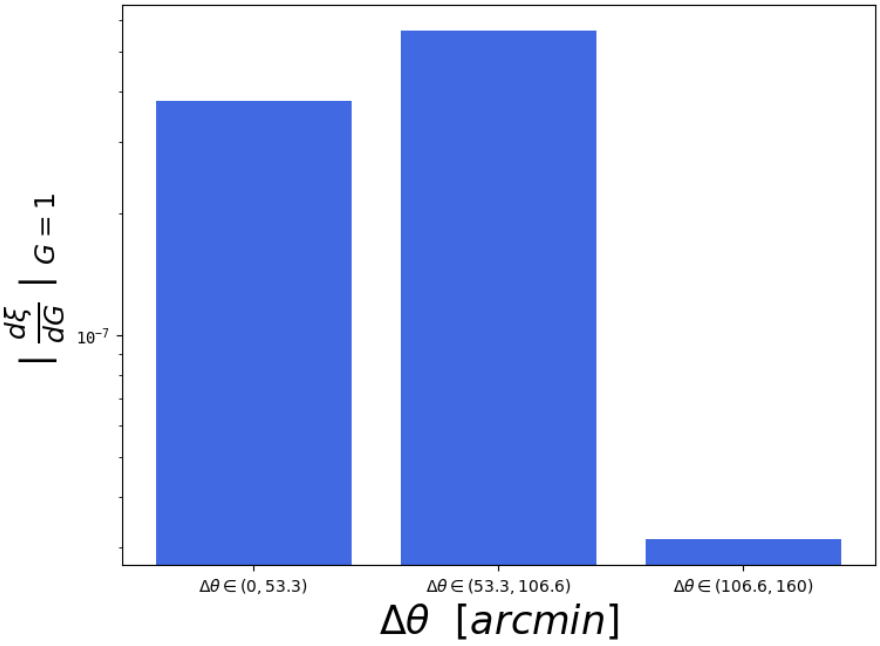



Figure 10: The gradient of the shear-shear correlation function $\xi$ with respect to the input model parameter $G$ at $G = 1$.

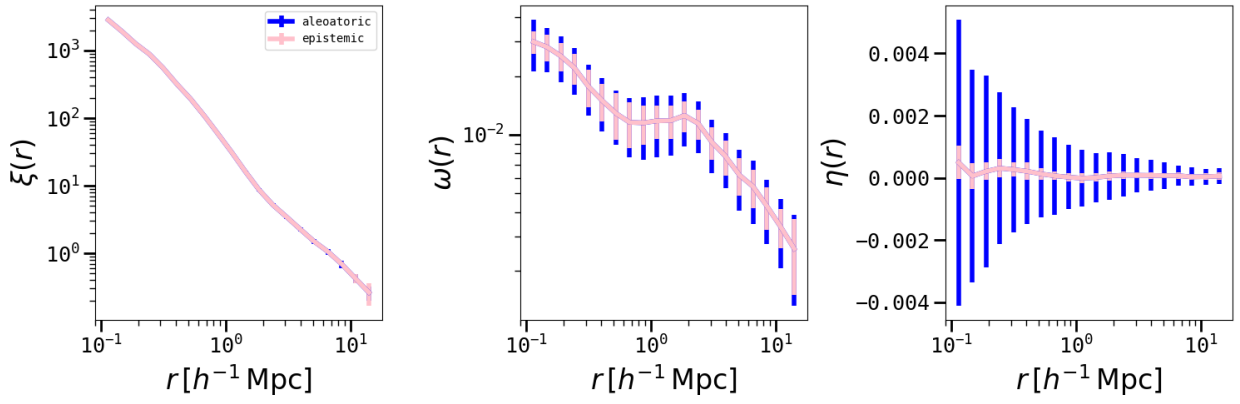
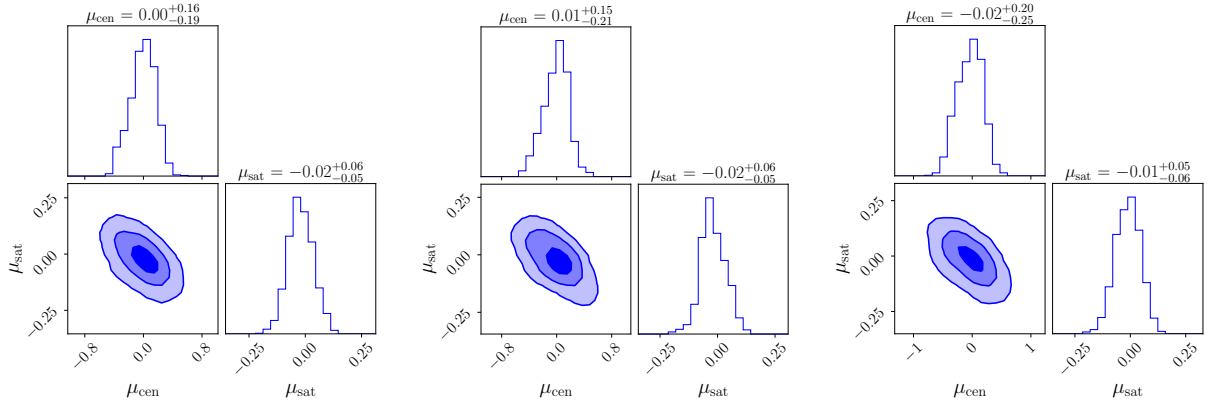
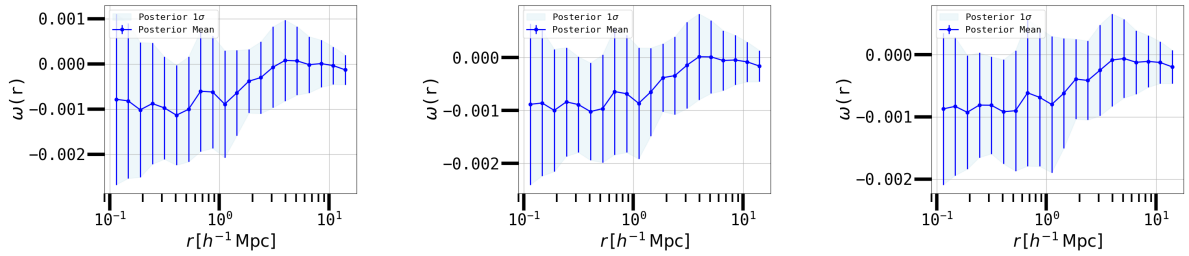



Figure 11: IAEmu predictions for a given initialization of the seven halo-based modeling parameters, including $\xi(r)$ on the left, $\omega(r)$ in the middle, and $\eta(r)$ on the right. For each of the twenty radial bins, the value of the correlation, as well as aleatoric and epistemic uncertainties, are provided. Log-log scaling is used to better see the dynamic range of the correlations and radial dependence.




(a) $\log M_* \geq 10.0$ (b) $\log M_* \geq 9.5$ (c) $\log M_* \geq 9.0$

Figure 12: Posterior constraints on $\mu_{cen}$ and $\mu_{sat}$ obtained via HMC for three different sets of HOD parameters. Each set roughly corresponds to a different stellar mass cutoff in galaxies. We show results for $\log M_* \geq 10.0$ on the left, $\log M_* \geq 9.5$ in the middle, and $\log M_* \geq 9.0$ on the right. Contours represent $1\sigma$, $2\sigma$, and $3\sigma$ confidence intervals.




(a) $\log M_* \geq 10.0$ (b) $\log M_* \geq 9.5$ (c) $\log M_* \geq 9.0$

Figure 13: Correlation functions sampled from the posterior distributions presented in 12. The solid represents the average correlation value in all bins from the 1000 samples of $(\mu_{cen}, \mu_{sat})$ pairs obtained via HMC. The error bar represents the standard deviation of the correlation value for that same sample.

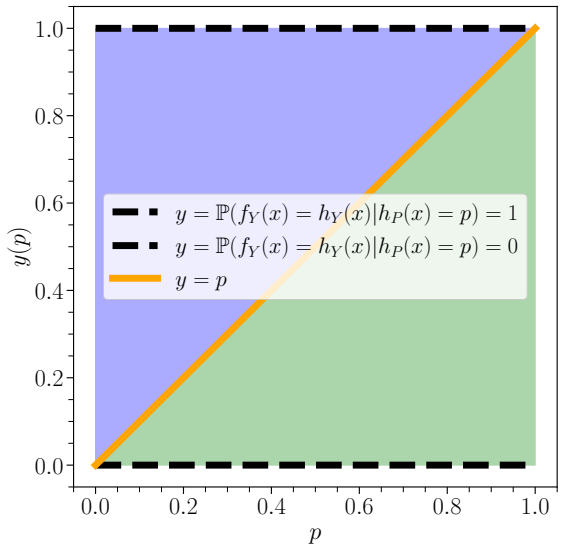



Figure 14: The relationship between $\mathbb{P}\left(f(x) = h_Y(x)|h_P(x) = p\right)$ and $p$ under different limiting values of accuracy. We see that calibration error is maximized when the accuracy is an extreme on the dotted black lines for any confidence $p$.

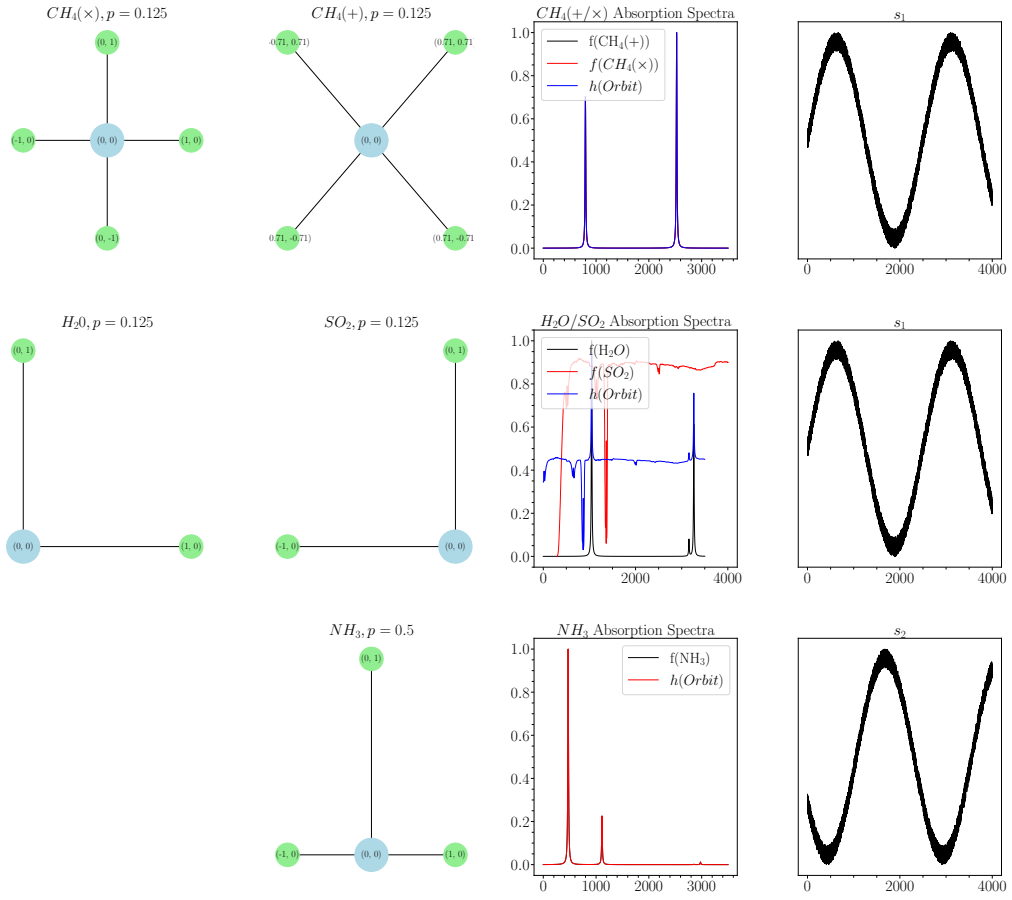



Figure 15: An example on how the ENCE upper bound behaves for an $E(2)$ invariant model class $h$ on a set of molecules producing absorption spectra. Each row contains an orbit, the absorption spectra specified by $f$ and the error minimizing $h$, and the predicted variance vector.

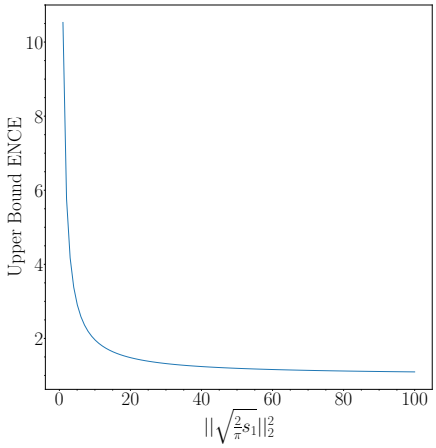



Figure 16: Upper bound on ENCE as a function of $\|\sqrt{\frac{2}{\pi}s_1}\|_2^2$ in Example 1.

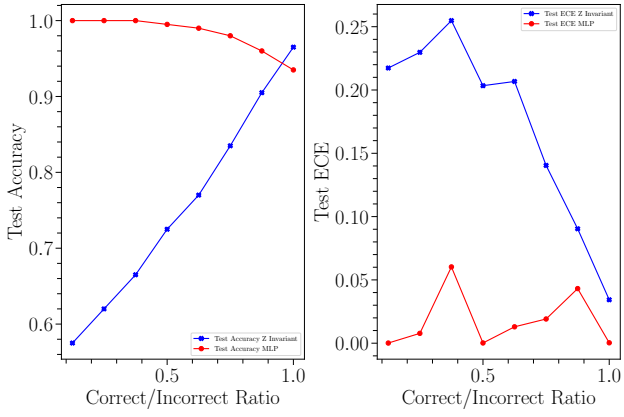



Figure 17: The left plot shows test accuracy for the z-invariant network (blue) and baseline unconstrained MLP (red) under different ratios of correct/incorrect ratios, ranging from 0% to 100% correctness. The right hand plot is the same but for ECE instead of accuracy.

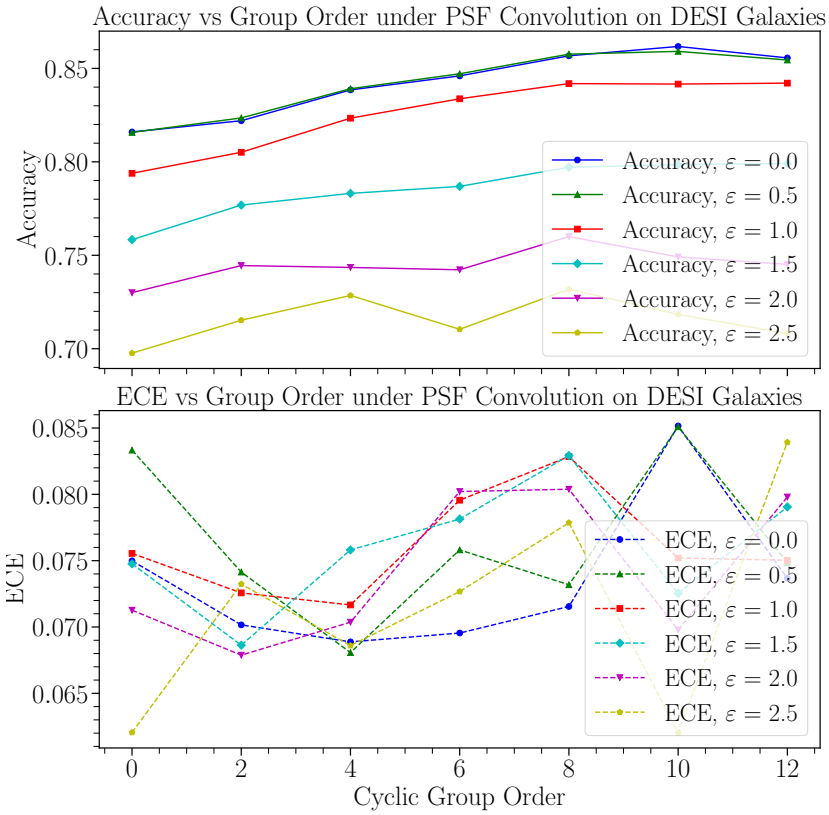



Figure 18: Accuracy and ECE vs Cyclic Group Order under PSF Convolution on DESI Galaxies.

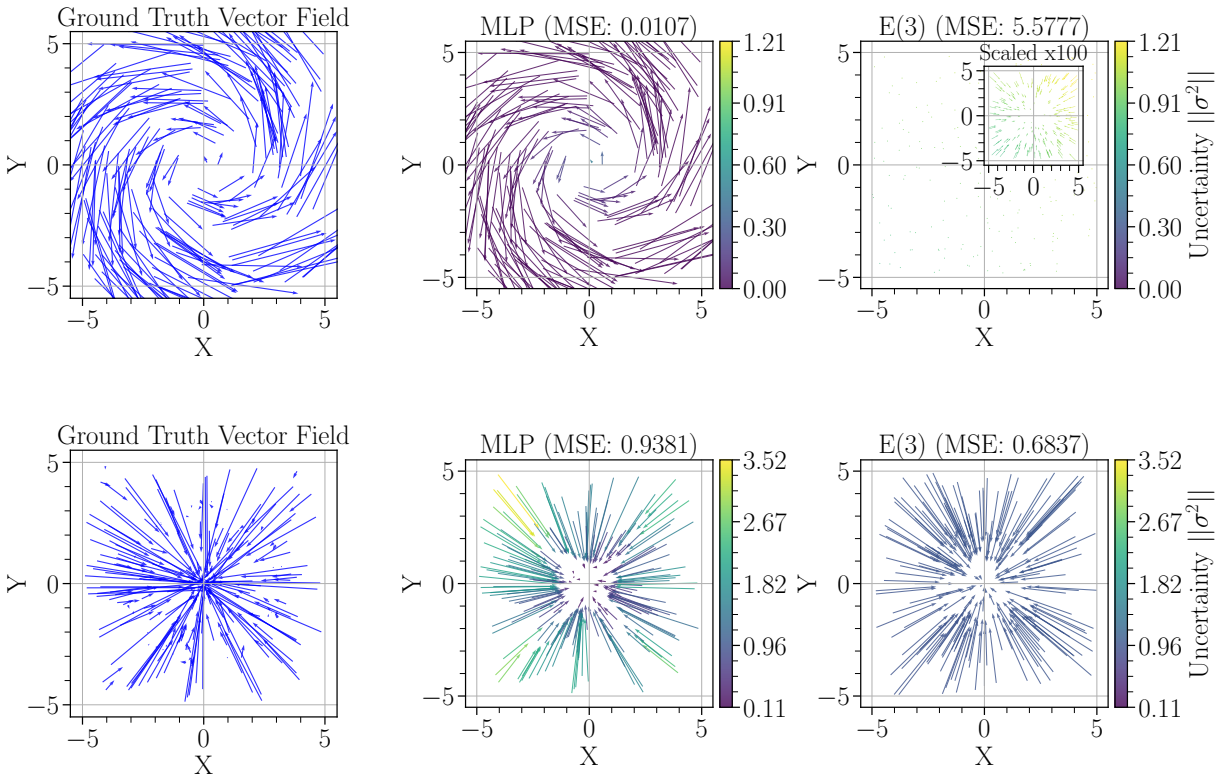



Figure 19: Vector regression results for the rotational and sinusoidal datasets (top and bottom respectively). For the model predictions in the middle and right columns, the color of the vector indicates the norm of the variance. The mean predictions for the $E(3)$ equivariant model on the rotational dataset are very small. The window in that image provides a 100 scaled up version of the image in order to see the behavior of the vectors.

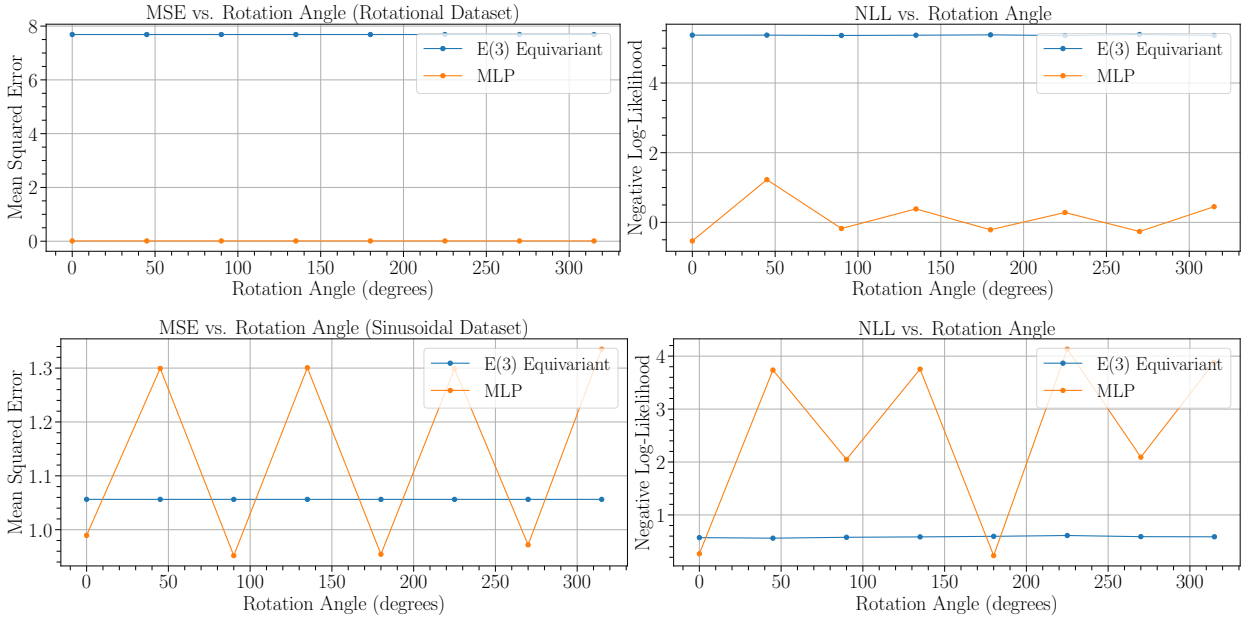



Figure 20: MSE and $\beta$-NLL losses for different rotation angle in the $xy-$plane for the rotational and sinusoidal datasets (top and bottom respectively).

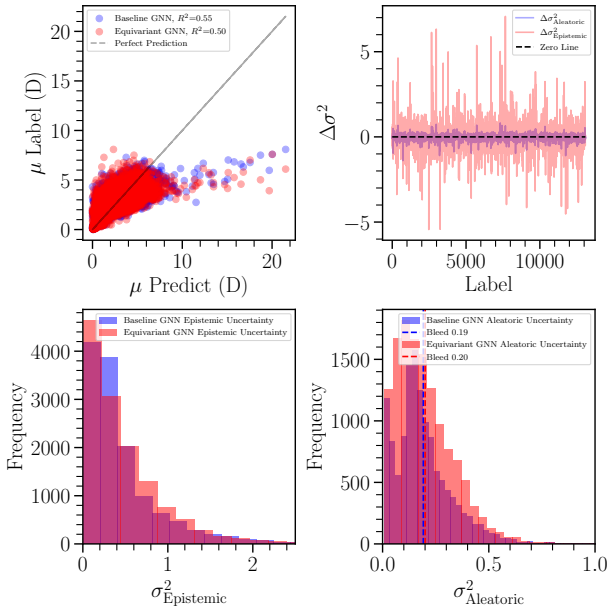



Figure 21: ***Top Left:*** Prediction versus label for both GIN and $E(3)-$Invariant models. ***Top Right:*** The difference between aleatoric and epistemic uncertainties between the two models for each label. ***Bottom Left:*** A distribution of the epistemic uncertainty predictions for the two models. ***Bottom Right:*** A distribution of the aleatoric uncertainty predictions for the two models.

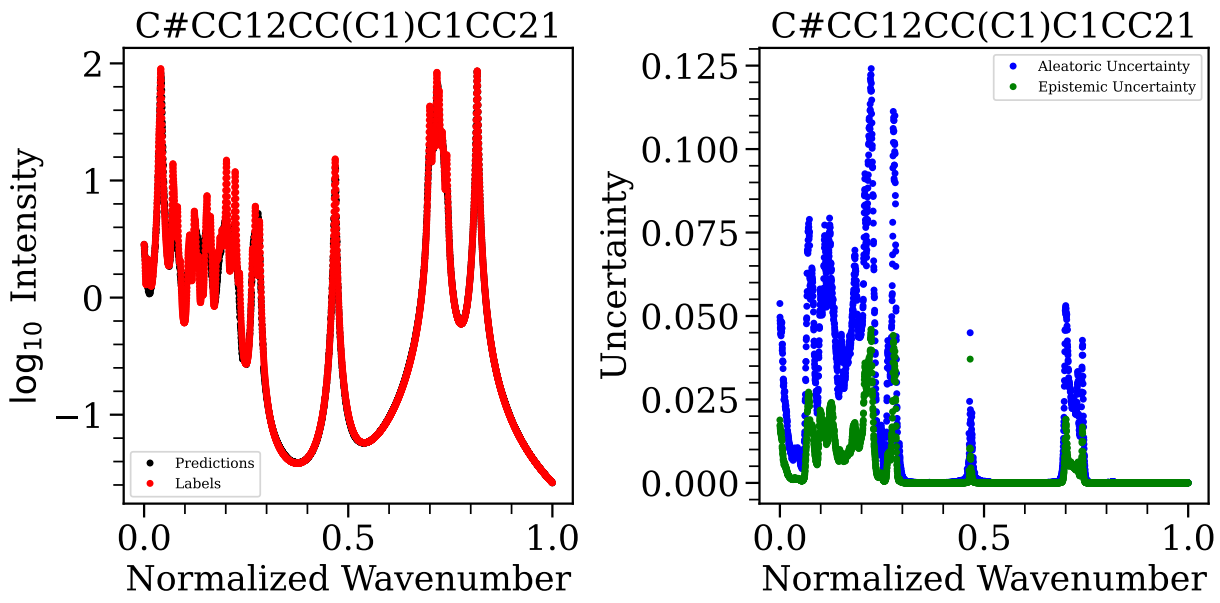



Figure 22: ***Left:*** Sample prediction vs ground truth spectra for the molecule given by SMILES (Weininger, 1988) string $C\#CC12CC(C1)C1CC21$. ***Right:*** The model's predicted aleatoric and epistemic uncertainties for each of the normalized wavenumbers.

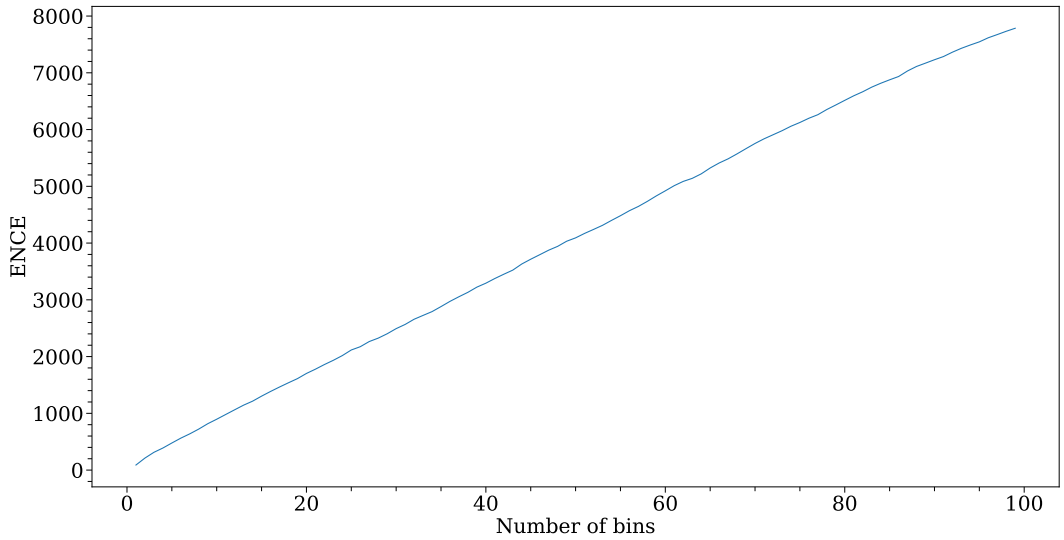



Figure 23: ENCE as a function of bins $N$.

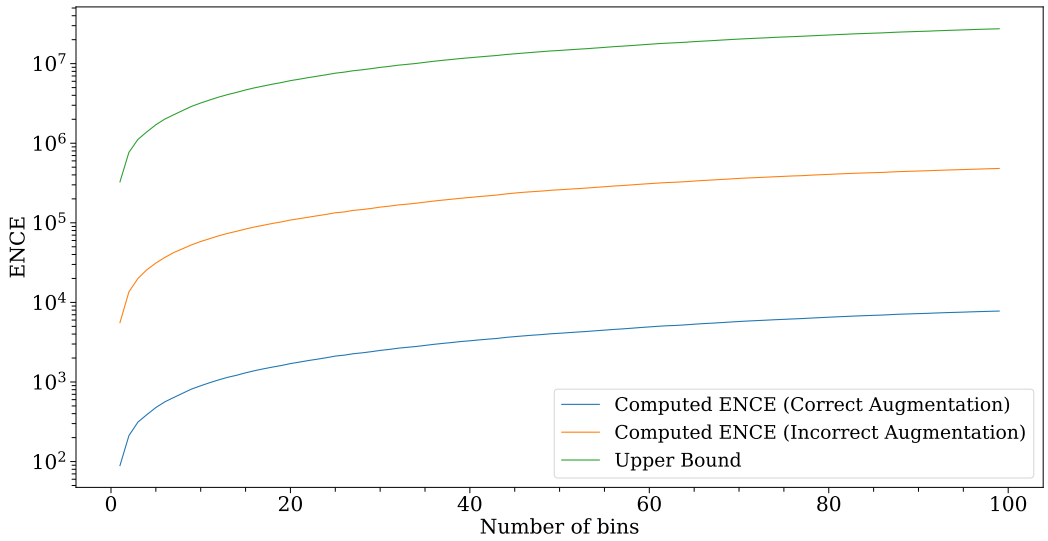



Figure 24: The Computed ENCE versus the theoretical upper bound.

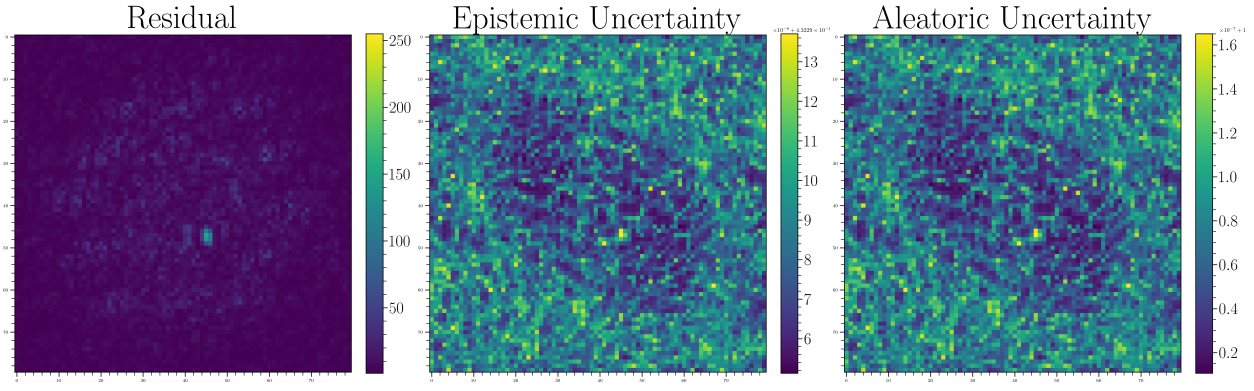



Figure 25: **Left:** The residual between image and PSF model. **Center:** The epistemic uncertainty. **Right:** The aleatoric uncertainty. The center color bar is in units of $\times 10^{-8} + 4.3229 \times 10^{-1}$ and the right color bar is in units of $\times 10^{-7} + 1$.

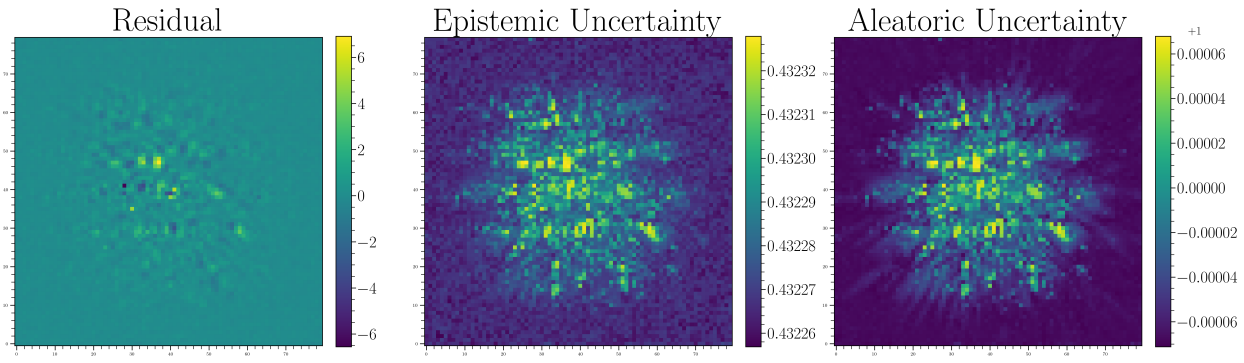



Figure 26: **Left:** The residual between image and PSF model. **Center:** The epistemic uncertainty. **Right:** The aleatoric uncertainty. As seen in the left panel, EDDO-UQ is able to successfully reveal the planet HIP65426b slightly up and to the left of the center of the image. Unlike pyKLIP, EDDO-UQ also produces estimates for the epistemic and aleatoric uncertainties.

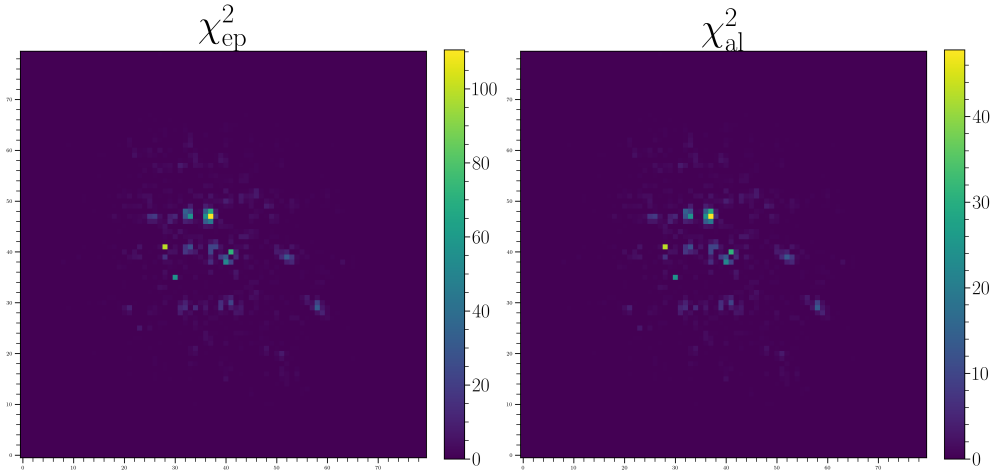



Figure 27: **Left:** The squared residuals weighted by the epistemic uncertainty. **Right:** The squared residuals weighted by the aleatoric uncertainty.

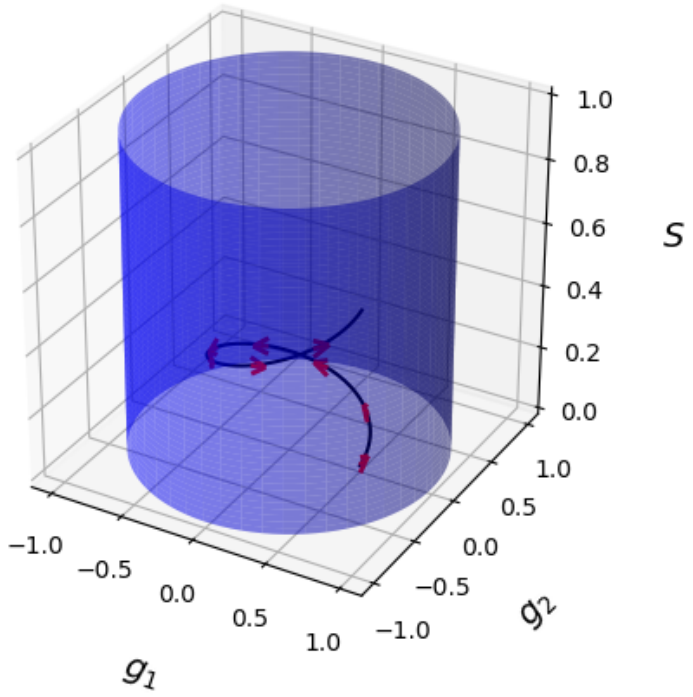



Figure 28: Parameter space for fitting an analytic profile, with sample iterations of the algorithm converging toward a learned $[s, g_1, g_2]$. Note that the cylinder extends upwards toward infinity but is bounded from below by 0.

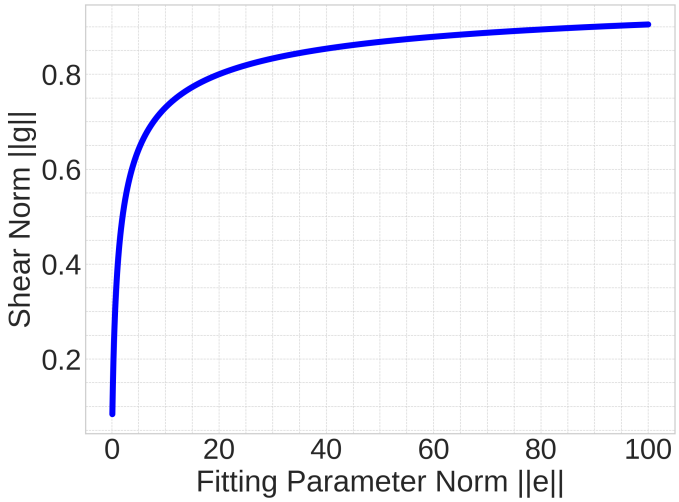



Figure 29: Reparameterization function. For any ellipticity vector $[e_1, e_2]$ the associated shear vector $[g_1, g_2]$ has a norm in $[0, 1)$.

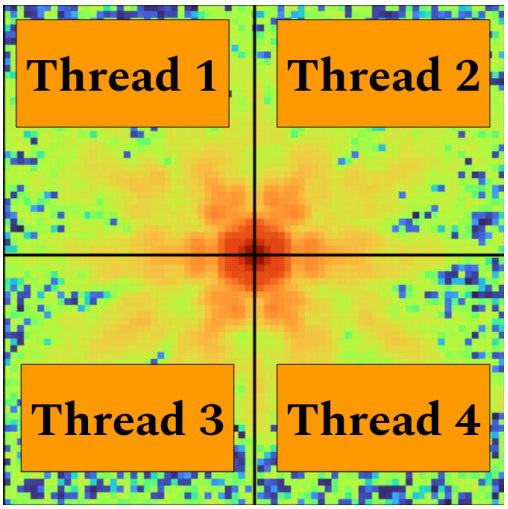



Figure 30: Pictoral representation of multi-threading in `ShOpt`. In this example, all of the pixels in the upper left of the pixel grid renderings are interpolated across the field of view in thread 1, all of the pixels in the upper right of the pixel grid renderings are interpolated across the field of view in thread 2, and so on.

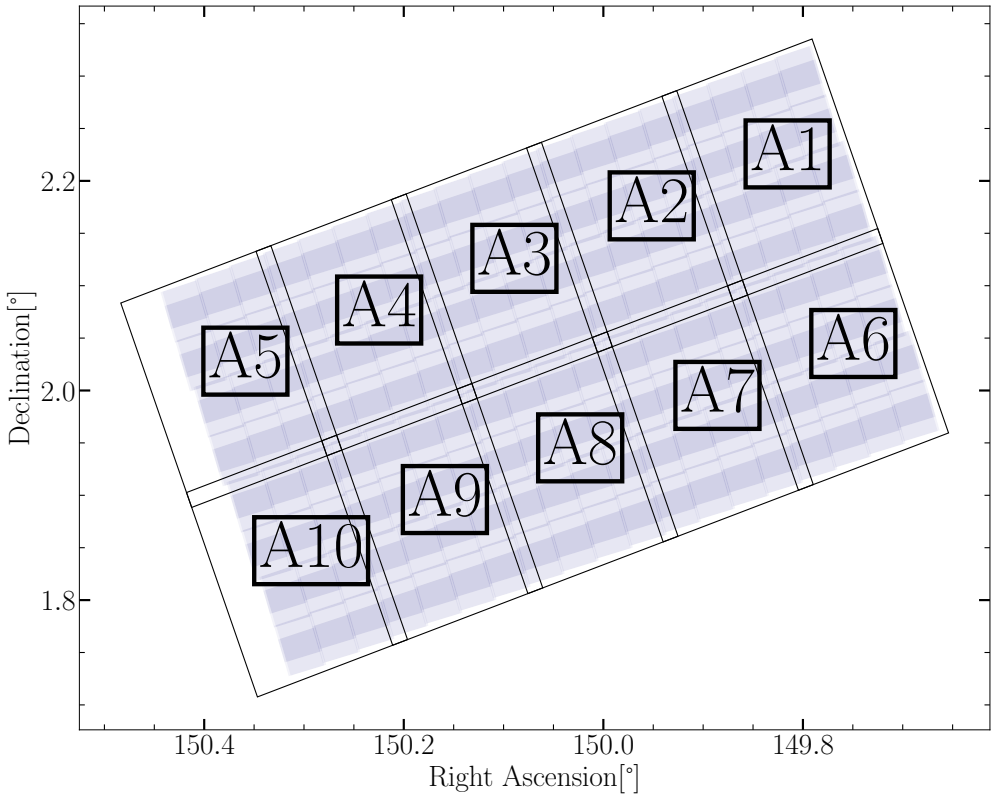



Figure 31: April data tiling scheme.

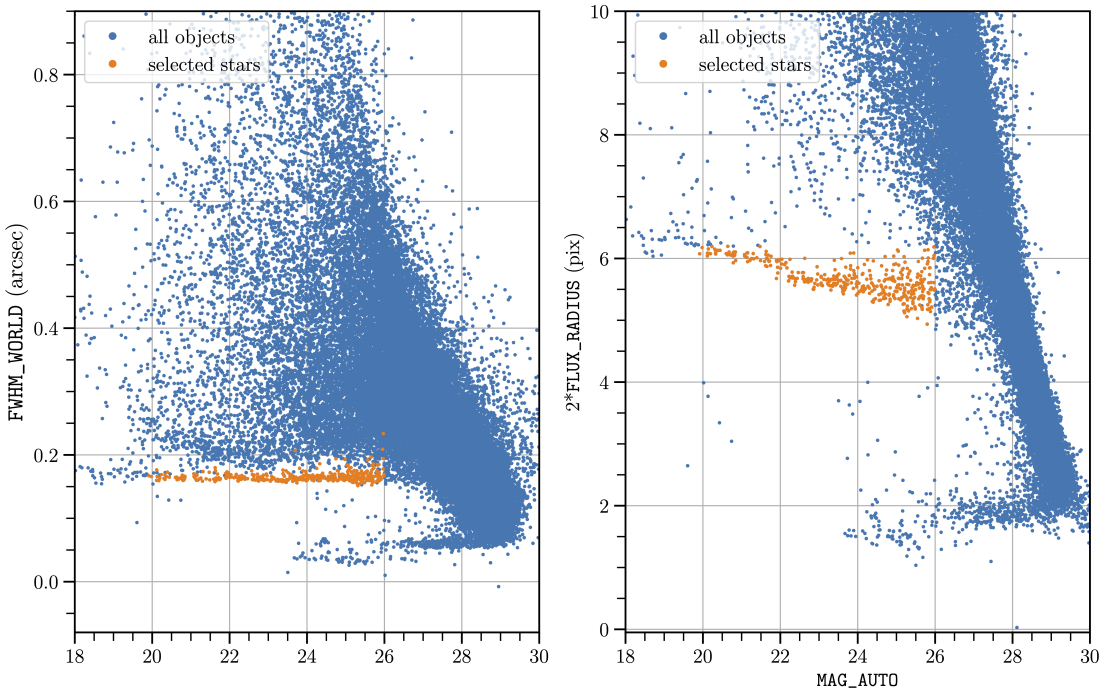



Figure 32: Size-magnitude diagram for objects detected in the A6 mosaic in F444W A6. Blue points represent all sources; orange points show stars selected by stellar locus parameter cuts.

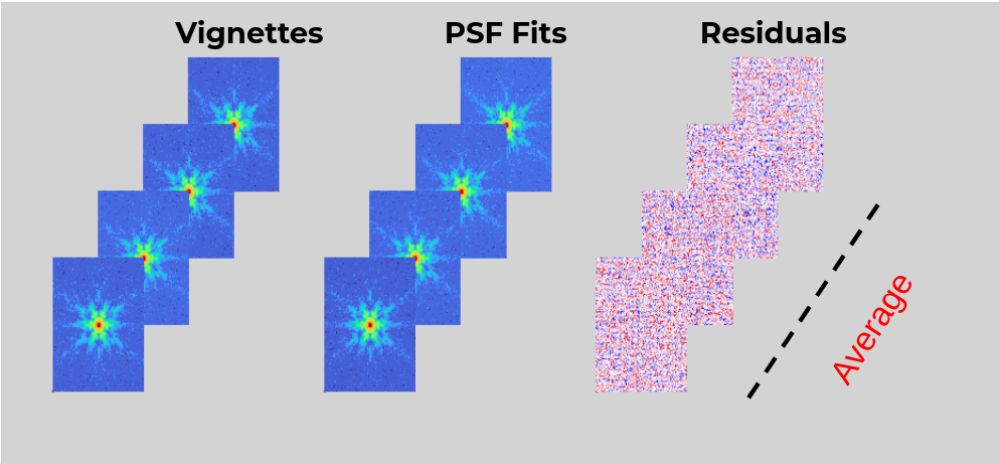



Figure 33: Schema illustrating the computation of summary statistics for PSF model assessment. Individual star-PSF model pairs are compared to produce residual images, which are then averaged into a mean residual image for the ensemble.

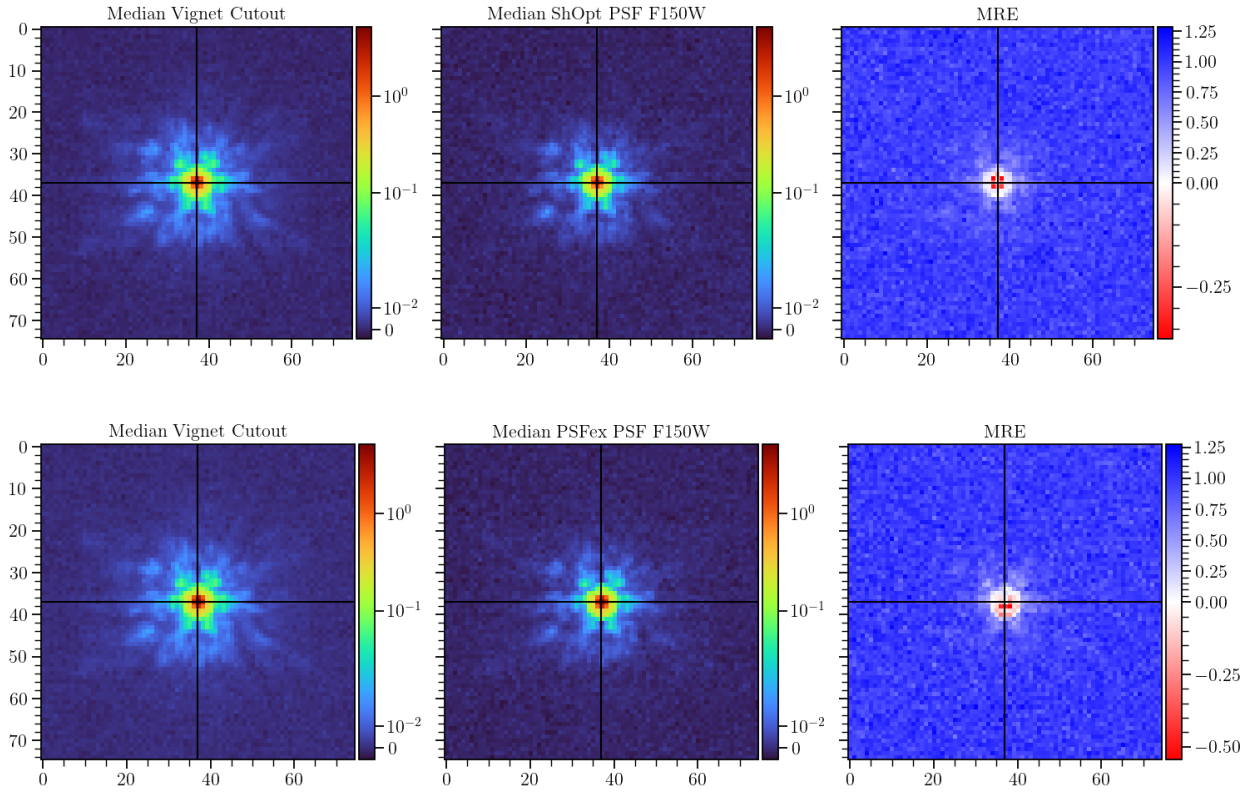



Figure 34: Mean relative error between MIRAGE input point source images and PSF models for the F277W filter. Left panels show the median MIRAGE cutout; center panels show the median PSF cutout. Right panels show the average relative error between the MIRAGE and PSF cutouts. Top two rows are `ShOpt`, bottom two rows are PSFEx. Color bars for left and center panels show pixel intensity values in units of MJy/sr. Color bars in the right panels show the (dimensionless) relative error. The MREs defined in Equation 92 are displayed in the titles of the mean residual images.

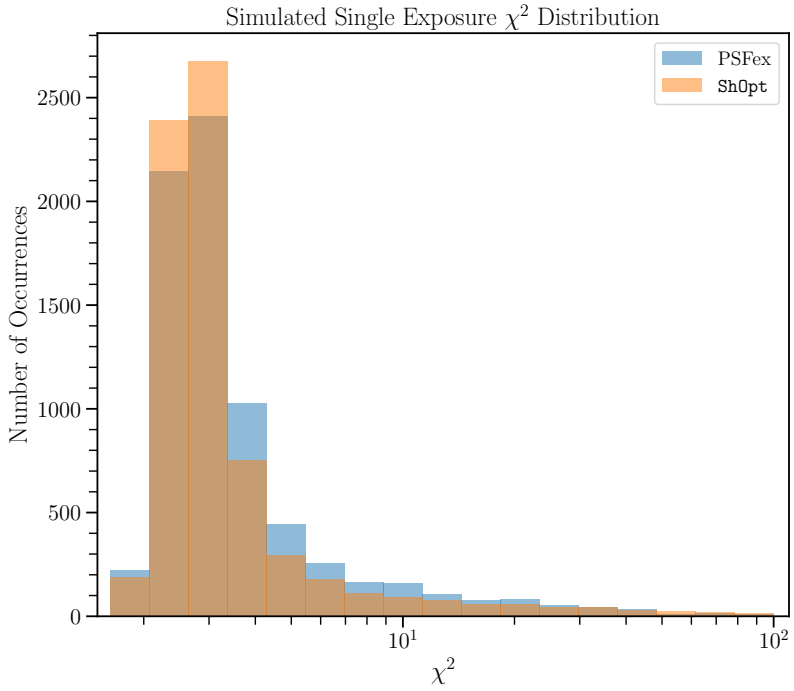



Figure 35: Distribution of $\chi^2_\nu$ for simulated single exposure images. Not shown are 42 PSFEx $\chi^2_\nu$ values greater than 100 and 100 `ShOpt` $\chi^2_\nu$ value greater than 100.

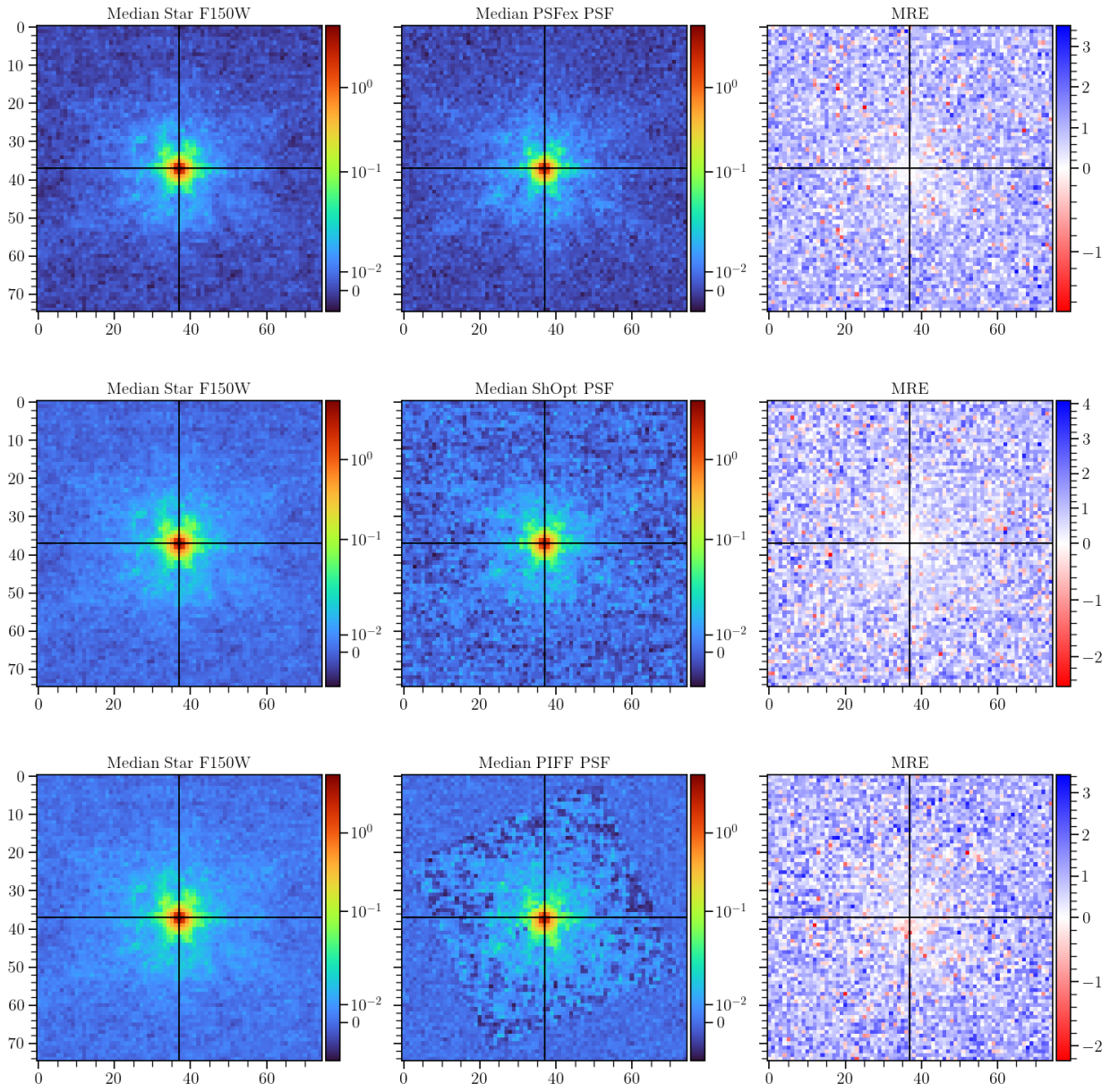



Figure 36: Evaluation of mean relative error between stars and PSF models for simulated mosaics in the F150W bandpass. Left panels show the median of the vignettes. Center panels show the median PSF cutout. Right panels show the average relative error between the vignette cutouts and the PSF cutouts. Top is PSFEx, middle is `ShOpt`, bottom is PIFF.

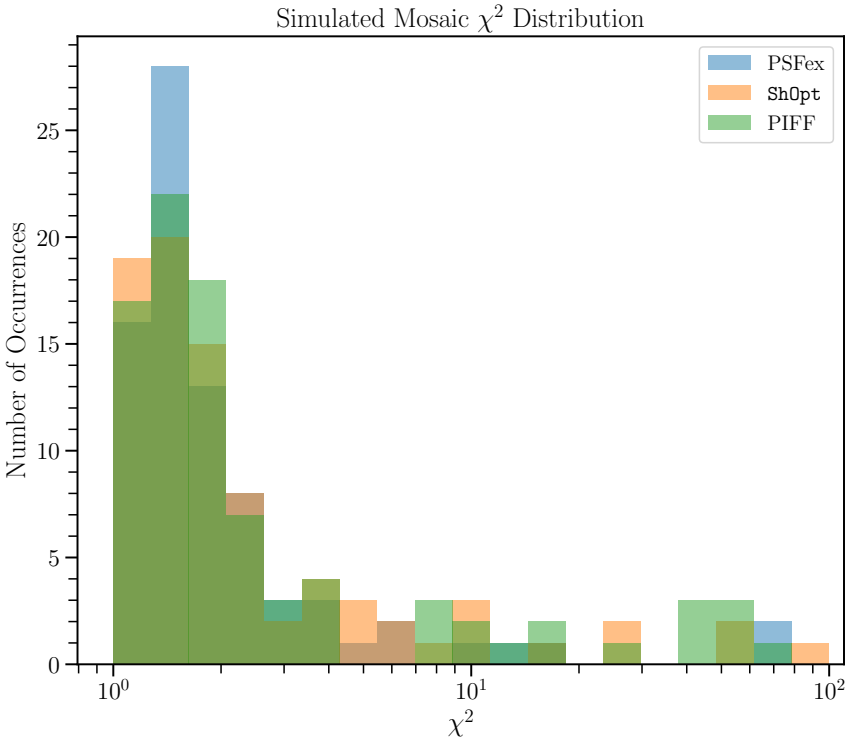



Figure 37: Distribution of $\chi^2_\nu$ for simulated data mosaics. PSFEx values are shown in blue, `ShOpt` values are shown in orange, and PIFF values are shown in green. For clarity, the plot excludes two PSFEx $\chi^2_\nu$ values greater than 100, one `ShOpt` $\chi^2_\nu$ value greater than 100, and two PIFF $\chi^2$ values greater than 100.

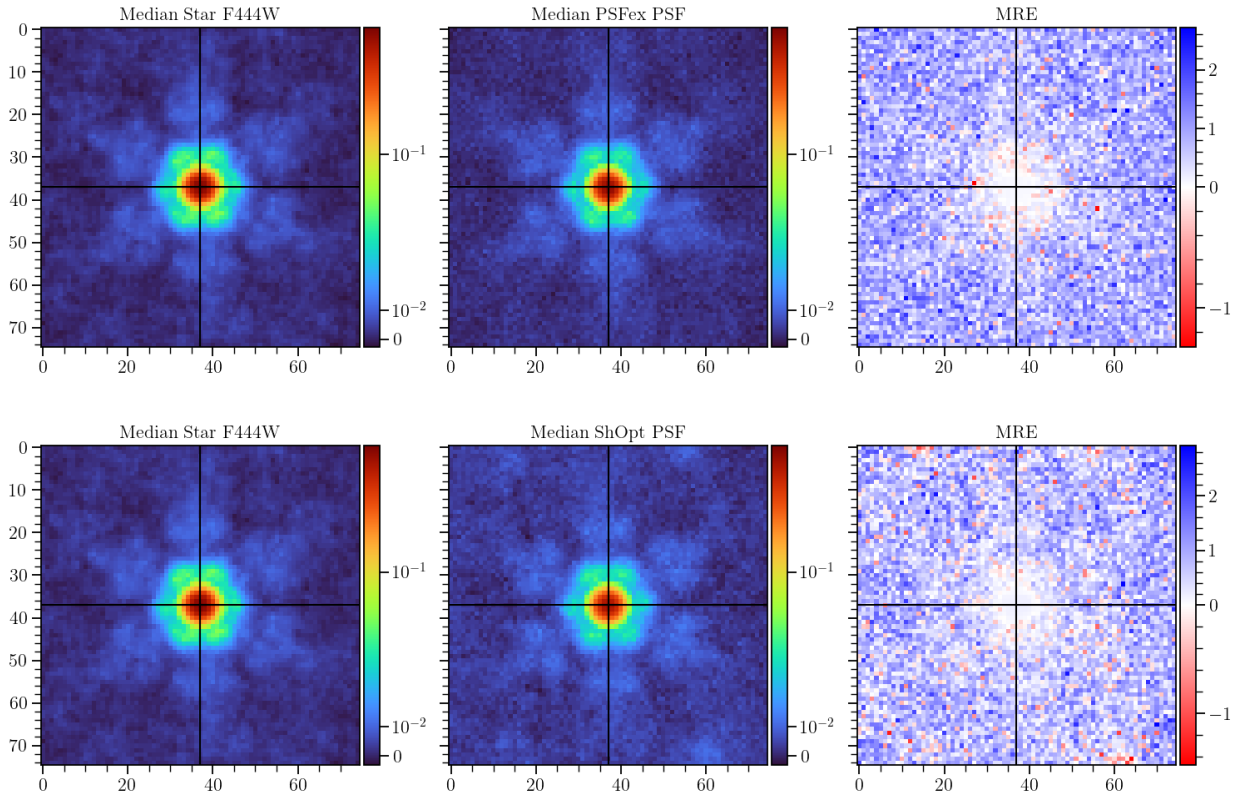



Figure 38: Evaluation of mean relative error between stars and PSF models for real data mosaics in the F444W bandpass. Left panels show the median star vignette; center panels show the median PSF model; right panels show the mean residuals of individual star-PSF vignettes. Top rows is PSFEx, bottom row is `ShOpt`.

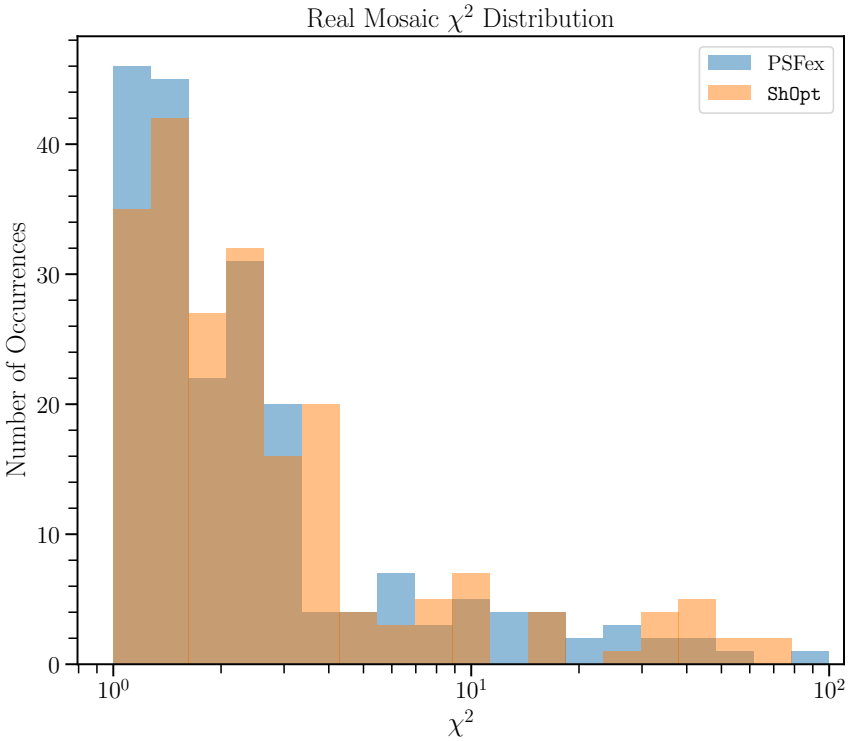



Figure 39: Distribution of $\chi^2_\nu$ for real data mosaics. Not shown are two PSFEx $\chi^2_\nu$ values greater than 100 and four ShOpt $\chi^2_\nu$ values greater than 100.

### 30.2 Size and Shape Analysis

Although the NIRCam PSFs are obviously not elliptical Gaussians, adaptive second moments are a common way of evaluating the quality of PSF fits (Hirata & Seljak, 2003; Mandelbaum et al., 2005), and so in this section we explore residuals in the second moment fits of stars and PSF models. Specifically, we use the the GalSim software's (Rowe et al., 2015) `FindAdaptiveMom` function to measure the size ($\sigma_{\text{HSM}}$) and shape ($(g1, g2)$) of the best-fit elliptical Gaussians of all validation stars and PSF models, then compute the average of the differences between the two. As demonstrated by Table 7, `ShOpt` is able to produce fits that are just as good if not better than PSFEx and PIFF across the different data sets and wavelengths. The only area where the `ShOpt` models struggled was with the shape adaptive moments on some of the simulated single exposures. Otherwise, `ShOpt` produced just as good if not better size and shape statistics compared to the other PSF fitters.

| | | | MSE | | |
|---|---|---|---|---|---|
| Data | Fitter | Wavelength | $\sigma_{\text{HSM}} \times 10^{-2}$ | $g_1 \times 10^{-2}$ | $g_2 \times 10^{-2}$ |
| Simulated Single Exposure | PSFEx | F115W | $0.134 \pm 0.138$ | $0.06 \pm 0.01$ | $0.17 \pm 0.05$ |
| Simulated Single Exposure | ShOpt | F115W | $0.248 \pm 0.032$ | $0.37 \pm 0.01$ | $0.23 \pm 0.01$ |
| Simulated Single Exposure | PSFEx | F150W | $0.46 \pm 0.12$ | $0.17 \pm 0.01$ | $0.24 \pm 0.01$ |
| Simulated Single Exposure | ShOpt | F150W | $0.56 \pm 0.13$ | $0.35 \pm 0.01$ | $0.23 \pm 0.01$ |
| Simulated Single Exposure | PSFEx | F277W | $2.48 \pm 4.30$ | $0.12 \pm 0.02$ | $0.10 \pm 0.02$ |
| Simulated Single Exposure | ShOpt | F277W | $15.3 \pm 10.90$ | $0.61 \pm 0.03$ | $1.68 \pm 0.06$ |
| Simulated Single Exposure | PSFEx | F444W | $1.75 \pm 0.67$ | $0.16 \pm 0.02$ | $0.29 \pm 0.04$ |
| Simulated Single Exposure | ShOpt | F444W | $3.46 \pm 0.89$ | $0.67 \pm 0.05$ | $1.12 \pm 0.09$ |
| Simulated Mosaics | PSFEx | F115W | $17.70 \pm 9.65$ | $1.63 \pm 0.63$ | $0.86 \pm 0.28$ |
| Simulated Mosaics | ShOpt | F115W | $18.44 \pm 10.60$ | $1.57 \pm 0.62$ | $0.83 \pm 0.26$ |
| Simulated Mosaics | PIFF | F115W | $18.80 \pm 10.90$ | $1.52 \pm 0.60$ | $0.93 \pm 0.31$ |
| Simulated Mosaics | PSFEx | F150W | $7.96 \pm 5.58$ | $0.72 \pm 0.51$ | $0.07 \pm 0.02$ |
| Simulated Mosaics | ShOpt | F150W | $7.35 \pm 5.00$ | $0.70 \pm 0.50$ | $0.07 \pm 0.02$ |
| Simulated Mosaics | PIFF | F150W | $7.53 \pm 5.35$ | $0.76 \pm 0.53$ | $0.07 \pm 0.02$ |
| Simulated Mosaics | PSFEx | F277W | $2.29 \pm 0.58$ | $0.42 \pm 0.13$ | $0.17 \pm 0.05$ |
| Simulated Mosaics | ShOpt | F277W | $2.83 \pm 0.83$ | $0.44 \pm 0.13$ | $0.16 \pm 0.04$ |
| Simulated Mosaics | PIFF | F277W | $5.19 \pm 0.81$ | $0.43 \pm 0.17$ | $0.13 \pm 0.04$ |
| Simulated Mosaics | PSFEx | F444W | $1.08 \pm 0.33$ | $0.17 \pm 0.05$ | $0.30 \pm 0.17$ |
| Simulated Mosaics | ShOpt | F444W | $0.98 \pm 0.27$ | $0.16 \pm 0.05$ | $0.32 \pm 0.18$ |
| Simulated Mosaics | PIFF | F444W | $2.28 \pm 0.31$ | $0.30 \pm 0.08$ | $0.30 \pm 0.17$ |
| April Mosaics | PSFEx | F115W | $0.67 \pm 0.10$ | $0.33 \pm 0.05$ | $0.19 \pm 0.05$ |
| April Mosaics | ShOpt | F115W | $0.78 \pm 0.10$ | $0.31 \pm 0.05$ | $0.20 \pm 0.05$ |
| April Mosaics | PSFEx | F150W | $0.38 \pm 0.06$ | $0.12 \pm 0.02$ | $0.04 \pm 0.01$ |
| April Mosaics | ShOpt | F150W | $0.56 \pm 0.06$ | $0.11 \pm 0.02$ | $0.03 \pm 0.00$ |
| April Mosaics | PSFEx | F277W | $1.19 \pm 0.23$ | $0.05 \pm 0.01$ | $0.04 \pm 0.01$ |
| April Mosaics | ShOpt | F277W | $0.90 \pm 0.13$ | $0.04 \pm 0.01$ | $0.03 \pm 0.01$ |
| April Mosaics | PSFEx | F444W | $2.02 \pm 1.49$ | $0.02 \pm 0.01$ | $0.01 \pm 0.00$ |
| April Mosaics | ShOpt | F444W | $2.00 \pm 1.63$ | $0.03 \pm 0.01$ | $0.01 \pm 0.00$ |

Table 7: The residuals in the adaptive second moments of stars and PSF models across all data sets used in this work.

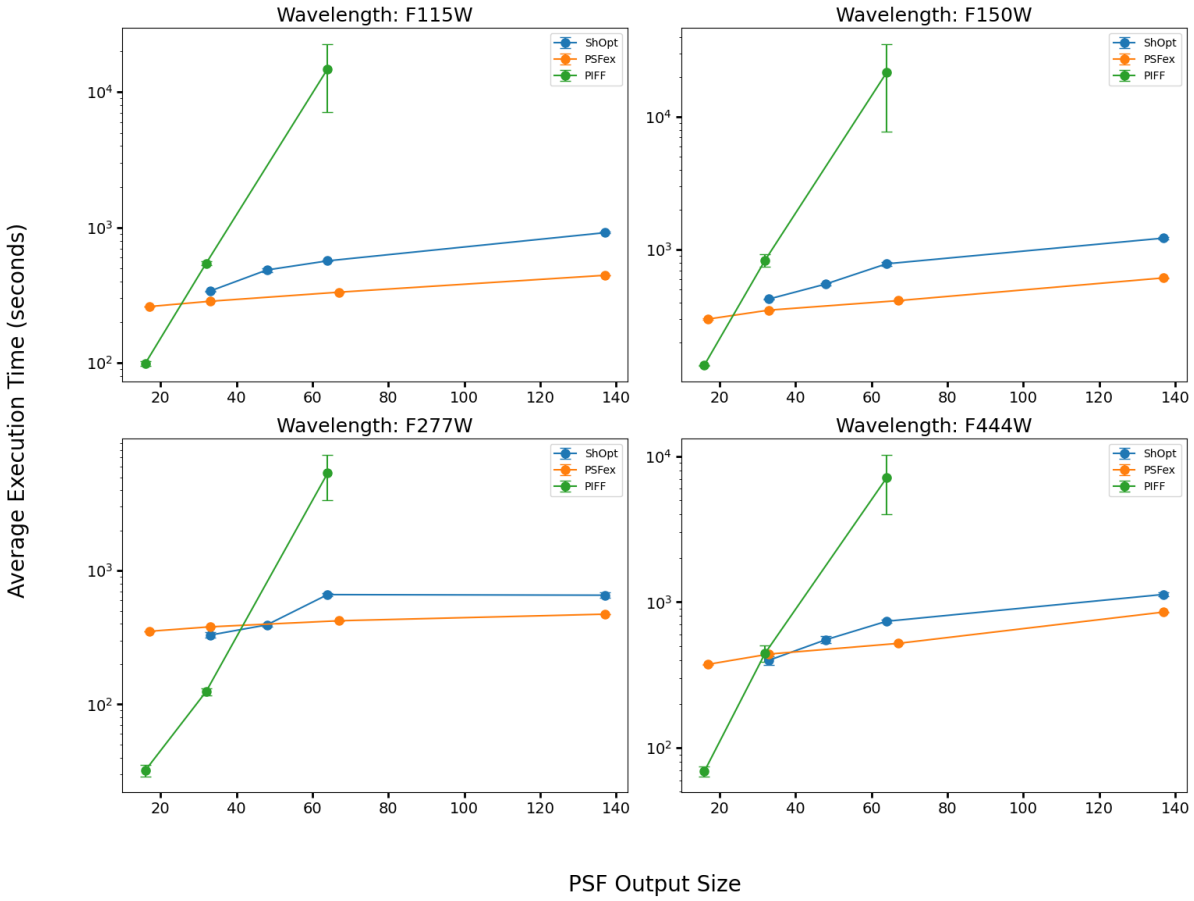



Figure 40: Average execution time as a function of the output size of the PSF sidelength plotted on a log scale.

### 30.3 Program Speed and Scalability

We investigate how the different PSF fitters handle the simulated mosaic data as the number of pixels in the star vignettes and the degree of the polynomial used to model spatial variations increase. Each PSF fitter is run multiple times to get an average time in seconds for pixel basis fits. For `ShOpt`, we set the size of the side length $n$ of the analytic fit stamp to be 30. That is, we use $30 \times 30$ pixels to fit the multivariate Gaussian. This is separate from the number of pixels we use for the pixel basis. We do not use any GPU compute power to accelerate the PSF fitters even though `ShOpt autoencoder` mode can be GPU-accelerated. All of these tests are run on the same hardware using Northeastern's Discovery Cluster on the zen2 CPUs (Northeastern University Research Computing, 2024). These tests aim to test our calculations in Section 28.

#### 30.3.1 Variation with vignette size

While the runtime for both `ShOpt` and PSFEx scales modestly as the output PSF size increases, the PIFF runtime does not. For larger PSF sizes, PIFF's average `BasisInterp` execution time is an order of magnitude longer than `ShOpt` and PSFEx. While `ShOpt` and PSFEx required approximately 5 to 10 minutes to process PSF sizes of $(137, 137)$ pixels for each of the four wavelengths, PIFF consistently took between 1 and 3 hours for PSF sizes of $(64, 64)$ (Figure 40). In the worst case, PIFF can take as much as an entire day to finish. While most PSF fitters had consistent completion times, PIFF exhibits greater variability for $(64, 64)$-pixel PSF output sizes, particularly with shorter wavelengths. It is worth noting that PIFF was developed using

DECam imaging, for which PSF sizes of $(17, 17)$ are sufficient. For those vignette sizes, PIFF is actually faster than the other PSF fitters, see Figure 40. It is also worth noting that PIFF contains alternatives to the `BasisInterp` algorithm benchmarked here and that these algorithms may be faster.

### 30.3.2 Polynomial Degree

As in the case of vignette size, while both `ShOpt` and PSFEx scale as the the degree of the spatial interpolation polynomial increases, PIFF does not. For a degree-1 interpolation, all three PSF fitters have similar performance for $(33, 33)$ PSF sizes. However, as we increase the degree to 2 and 3, PIFF takes an order of magnitude longer on average to execution, see Figure 41. For larger degree sizes, PIFF often did not converge to a PSF model with $4\sigma$ confidence after its maximum of 30 iterations. `ShOpt` and PSFEx maintain a consistent runtime of about 5 minutes, whereas PIFF experiences a dramatic increase in runtime, going up to hours for short wavelengths with higher degrees of the interpolating polynomial, and by around thirty minutes for longer wavelengths.

## 31 Discussion and Conclusions

We have presented `ShOpt`, a new empirical PSF characterization tool, and with it, a methodology for benchmarking the accuracy and computational efficiency of PSF fitting software.

In our assessment of PSF model quality, we produced a series of mean residual images, including normalized mean error, mean absolute error, and chi-squared error. We further condense this information into aggregate statistics that quantify the pixel-level discrepancies between the modeled and actual PSFs across the ensemble of stars being modeled. We supplement these statistics with second moment HSM fit statistics because of their wide use in the literature.

Though convenient, single-number figures of merit can mask discrepancies between a PSF model and the true PSF. Accordingly, we recommend a holistic approach for evaluating the quality of a particular PSF model, using both aggregate statistics and mean residual images. Among the various single-number figures of merit we examined, the average and median chi-squared values were the most illustrative of mismatched PSF models. Additionally, the distributions of these statistics are usually indicative of the performance of the PSF modeling.

In Section 28, we predicted that `ShOpt`'s algorithmic design would allow it to scale as the PSF model size and the degree of the interpolating polynomial increased; this assertion was borne out in our speed testing. Our analysis does not distinguish between the contributions of architectural choices and the Julia language itself to `ShOpt`'s fast execution time. Regardless, we find that `ShOpt` delivers speed performance on par with PSFEx, with only marginal differences in processing times for PSF models $(137, 137)$ pixels in size (large enough to enclose most of the NIRCam PSF). `ShOpt`'s competitive speed comes from a combination of multithreading, imposed geometric constraints, the implementation of the LBFGS algorithm, and a thorough data cleaning pipeline. Conversely, PIFF—optimized for the larger pixel scale of the Dark Energy Camera—does not execute in a reasonable timeframe even for PSF sizes that are $(64, 64)$ pixels in size.

In its current state, `ShOpt` produces an excellent model of the NIRCam PSF for all three data sets, and is able to produce these models extremely fast.

Several enhancements are planned for `ShOpt` that may improve the quality of its PSF models. For example, `ShOpt` builds models strictly in the native pixel scale of the image; implementing super- or sub-sampling of the image pixel scale would align `ShOpt` with the standards of other PSF software. A more advanced PCA method for PSF reconstruction, proposed in Lin et al. (2023), is slated for future integration in our `PCA` mode. There is also a method of non-negative PCA outlined in (Bro & De Jong, 1997) that may be useful to incorporate. These upgrades, among others, will be featured in future `ShOpt` releases.

`ShOpt` incorporates several innovative techniques that could benefit other PSF modeling efforts. These include leveraging manifold properties when fitting analytic profiles; using low-dimensional reconstructions for pixel-basis fits; and pixel-by-pixel parallelization during fitting of spatial variation. It has demonstrated

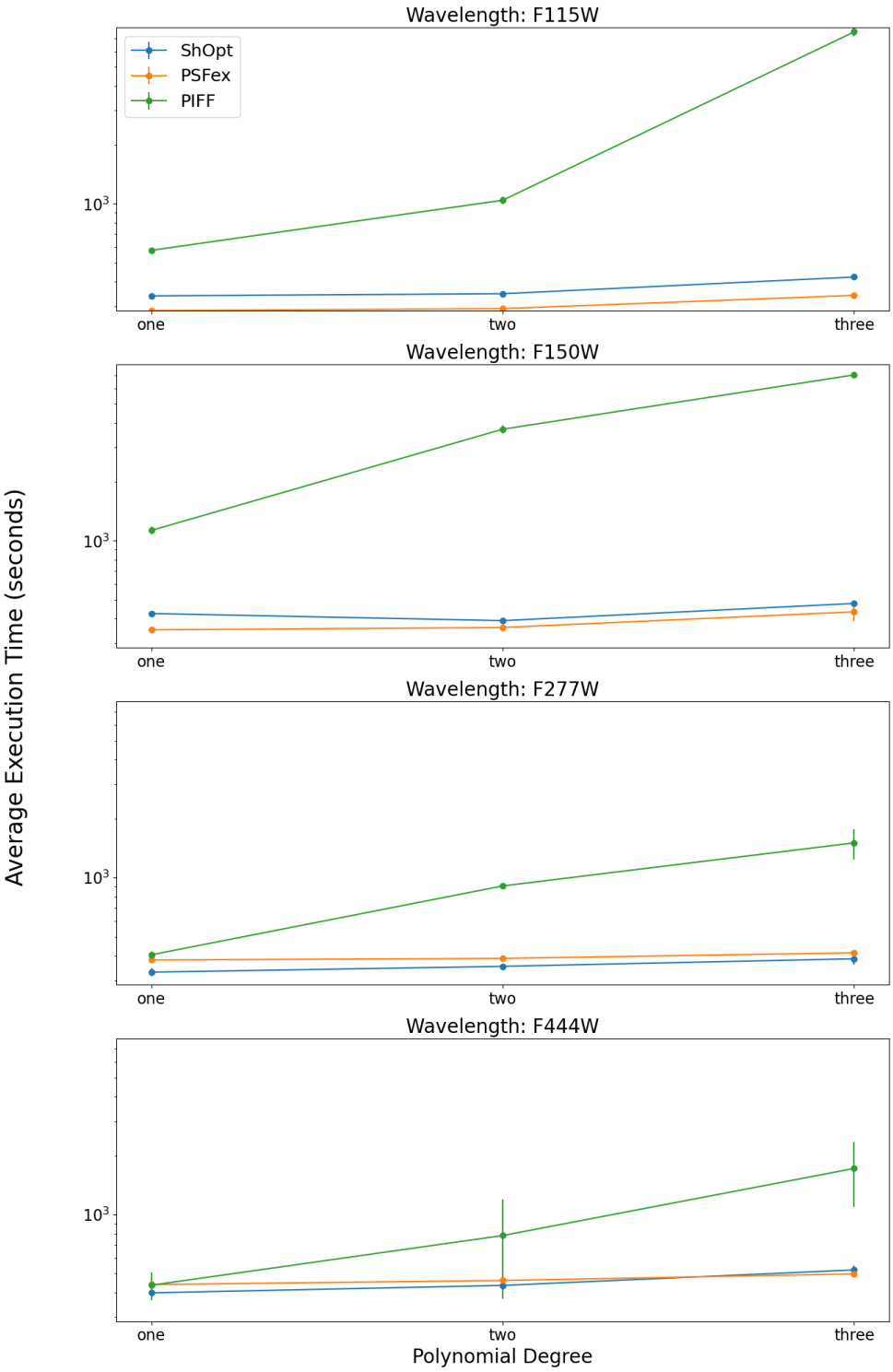



Figure 41: Average execution time as a function of the degree of the interpolating polynomial on $(33, 33)$ PSF models plotted on a log scale.

the ability to bridge the speed of PSFEx with the advances of PIFF, making it a viable PSF fitter for next-generation astronomical observatories.

# Part VI

# ShOpt.jl: A Julia Package of Point-spread Function Characterization of JWST NIRCam Data

## 32 Introduction

In this chapter, we provide a brief technical report that describes `ShOpt` in the broader picture of open source software.

## 33 Summary

When astronomers capture images of the night sky, several factors – ranging from diffraction and optical aberrations to atmospheric turbulence and telescope jitter – affect the incoming light. The resulting distortion and blurring are summarized in the image's point spread function (PSF), the response of an optical system to an idealized point source. The PSF can obscure or even mimic the astronomical signal of interest, making its accurate characterization essential. By effectively modeling the PSF, we can predict image distortions at any location and proceed to deconvolve the PSF, ultimately reconstructing distortion-free images.

The PSF characterization methods used by astronomers fall into two main classes: forward-modeling approaches, which use physical optics propagation based on models of the optics, and empirical approaches, which use stars as fixed points to model and interpolate the PSF across the rest of the image. (Stars are essentially point sources before their light passes through the atmosphere (when observing from the ground) and telescope, so the shape and size of their surface brightness profiles define the PSF at that location.) Empirical PSF characterization proceeds by first cataloging the observed stars, separating the catalog into validation and training samples, and interpolating the training stars across the field of view of the camera. After training, the PSF model can be validated by comparing the reserved stars to the PSF model's prediction.

Shear Optimization with `ShOpt.jl` introduces modern techniques, tailored to James Webb Space Telescope (JWST) NIRCam imaging, for empirical PSF characterization across the field of view. ShOpt has two modes of operation: approximating stars with analytic profiles, and a more realistic pixel-level representation. Both modes take as input a catalog with image cutouts – or "vignettes" – of the stars targeted for analysis.

## 34 Statement of need

Empirical PSF characterization tools like Point Spread Function Extractor (PSFex, Bertin, 2011) and Point Spread Functions in the Full Field of View (PIFF, Jarvis et al., 2020) are widely popular in astrophysics. However, the quality of PIFF and PSFEx models tends to be quite sensitive to the parameter values used to run the software, with optimization sometimes relying on brute-force guess-and-check runs. PIFF is also notably inefficient for large, well-sampled images, taking hours in the worst cases. The JWST's Near Infrared Camera (NIRCam) offers vast scientific opportunities (e.g. Casey, Kartaltepe, et al., 2023); at the same time, this unprecedented data brings new challenges for PSF modeling:

(1) Analytic functions like Gaussians are incomplete descriptions of the NIRCam PSF, as evident from Figures 1 and 2. Figure 1 shows that to get a good PSF model we need to model a large dynamic

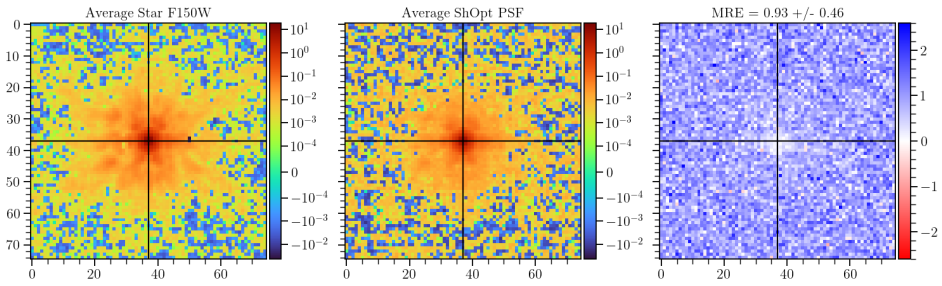



Figure 42: The plot on the left shows the average cutout of all stars in a supplied catalog. The plot in the middle shows the average point spread function model for each star. The plot on the right shows the average normalized error between the observed star cutouts and the point spread function model.

range over a large box size. Figure 2 illustrates that the residuals between an idealized forward model and a Gaussian approximation can be quite severe. This calls for well-thought-out, non-parametric modeling and diagnostic tools that can capture the full dynamic range of the NIRCam PSF. `ShOpt` provides these models and diagnostics out of the box.

(2) The NIRCam detectors have pixel scales of 0.03 (short wavelength channel) and 0.06 (long wavelength channel) arcseconds per pixel (Beichman et al., 2012; Rieke et al., 2003, 2005). At these pixel scales, star vignettes need to be at least 131 by 131 pixels across to fully capture the wings of the PSFs (4-5 arcseconds). These vignette sizes are 3-5 times larger than the ones used in surveys such as DES (Jarvis et al., 2020) and SuperBIT (McCleary et al., 2023) and force us to evaluate how well existing PSF fitters scale to this size. `ShOpt` has been designed for computational efficiency and aims to meet the requirements of detectors like NIRCam.

`ShOpt` bridges the speed of `PSFex` with the features of `PIFF` using fewer configurable hyperparameters. `ShOpt` employs a number of techniques to optimize the speed of the program. First and foremost, `ShOpt` is equipped with support for multithreading. Polynomial interpolation is used for handling PSF variations across the field of view. The polynomials given to each basis element of the PSF are independent of one another and therefore can be distributed to different CPU threads to be run in parallel. `ShOpt` also introduces new methods for fitting both analytic profiles and pixel based profiles. If an analytic profile is used to model the PSF, then there are 3 basis elements parameterizing the model. We choose 2D elliptical Gaussians for these analytic profiles because they are cheap to compute. Moreover, we use the Limited Memory Broyden–Fletcher–Goldfarb–Shanno algorithm (LBFGS) to find these paramters. This is faster but more memory intensive than Conjugate Gradient, the algorithm used in `PIFF`. Moreover, the 3 basis elements are constrained to the manifold $B_2(r) \times \mathbb{R}_+$, where $B_2(r) = \{(x, y) : x^2 + y^2 < 1\}$ and $\mathbb{R}_+ = \{x : x > 0\}$. We constructed a function that maps any point in $\mathbb{R}^3$ into $B_2(r) \times \mathbb{R}_+$. The LBFGS algorithm uses successive iterations to converge to a solution for the 3 basis elements, and so we use this function to ensure that our update steps in LBFGS do not leave the constraint. For the pixel basis, both `PIFF` and `PSFex` approximate the PSF by minimizing the reduced $\chi^2$ between a grid of pixels and a star vignette. PCA can quickly achieve the same purpose of approximating the input vignette without overfitting to background noise. We also provide an autoencoder mode, which uses deep learning to reconstruct the image. The weights and biases are not reset between stars, so the knowledge of how to reconstruct one star is transfered to the next. This in turn leads to fewer training iterations. Finally, `ShOpt` is written in Julia. Julia uses a just in time compiler which makes it faster than intepreted languages such as Python and has shown to be a good choice for performance critical code (Stanitzki & Strube, 2021).

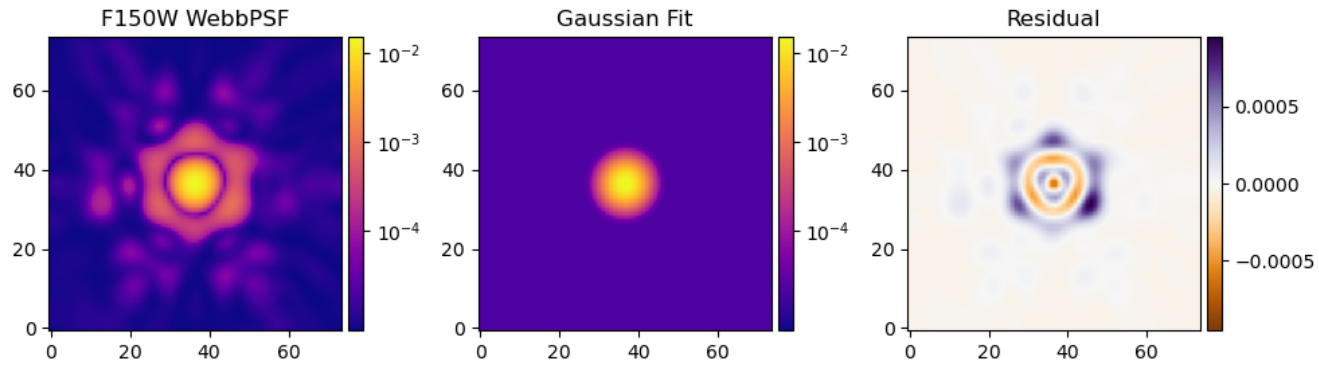



Figure 43: The plot on the left shows an idealized forward model of the NIRCam F150W PSF made using WebbPSF. The middle shows a Gaussian approximation to the PSF. The right shows the residual between the WebbPSF model and the Gaussian approximation.

## 35 State of the Field

The JWST captures images at high resolution and at wavelengths of light that have been previously unexplored (Gardner et al., 2006). With these images we are seeing farther into the early universe than we ever have before. The complex optics of JWST coupled with our desire to get the most out of the incredible data have presented the community with new set of challenges that current software does not address: the need to create high-fidelity models of the point spread function for diffraction-limited images across a wide range of wavelengths. `ShOpt.jl` is one attempt to solve this challenge and other teams are approaching the issue in complementary ways, as described below.

A number of JWST PSF libraries already exist, using both empirical and forward-modeling approaches. We describe them here and draw attention to their strengths and weaknesses to further motivate the development of `ShOpt.jl`. As described in the statement of need, `PSFex` was one of the first precise and general purpose tools used for empirical PSF fitting. However, the Dark Energy Survey collaboration reported small but noticeable discrepancies between the sizes of `PSFex` models and the sizes of observed stars. They also reported ripple-like patterns in the spatial variation of star-PSF residuals across the field of view (Jarvis et al., 2020), which they attributed to the astrometric distortion solutions for the Dark Energy Camera.

These findings motivated the Dark Energy Survey's development of `PIFF` (Point Spread Functions in the Full Field of View). `PIFF` works in sky coordinates on the focal plane, as opposed to image pixel coordinates used in `PSFex`, which minimized the ripple patterns in the star-PSF residuals and the PSF model size bias. (Based on the DES findings, `ShOpt` also works directly in sky coordinates.) `PIFF` is written in Python, a language with a large infrastructure for astronomical data analysis, for example Astropy (Astropy Collaboration et al., 2022) and Galsim (Rowe et al., 2015). The choice of language makes `PIFF` software more accessible to programmers in the astrophysics community than `PSFex`, which was first written in `C` in 2007 and much less approachable for a community of open source developers. One of the motivations of `ShOpt` was to write astrophysics specific software in `Julia`, because `Julia` provides a good balance of readability and speed with

its high-level functional paradigm and just-in-time compiler (Stanitzki & Strube, 2021). Julia ranks behind Python, IDL, Matlab, and Fortran in full-text mentions in astronomical literature (Collaboration et al., 2022). We are optimistic that `ShOpt` will demonstrate that Julia is an appealing choice for programming in astronomy despite its low adoption to date. There is also recent work on using `PSFr` for PSF reconstructions that has been applied to JWST data (Ding et al., 2022; Merlin et al., 2022; Santini et al., 2023; Yang et al., 2022). Similar to `ShOpt`, `PSFr` begins its PSF modeling by stacking stars to form an initial guess. However, rather than using polynomial interpolation to address spatial variations, `PSFr` employs an iterative process of shifting the pixels in its PSF model. This process continues until the model can adequately represent all of the stars in the catalog. Finally, the PSF fitter `STARRED` (Michalewicz et al., 2023) has been shown to produce PSF models competitive with `PSFr` and `PSFex`. Like `ShOpt`, `STARRED` puts an emphasis on computational efficiency and uses the JAX package to achieve just-in-time compilation. There is future work to be done to benchmarking all of these empirical approaches on JWST NIRCam data.

While WebbPSF provides highly precise forward models of the JWST PSF, these models are defined for single-epoch exposures (Perrin et al., 2012, 2014). Much of the NIRCam science is accomplished with image mosaics – essentially, the combination of single exposure detector images into a larger, deeper image. The rotation of the camera between exposures, the astrometric transformations and resampling of images before their combination into a mosaic, and the mosaic's large area all make the application of WebbPSF models to mosaics a non-trivial procedure. This has been done quite effectively (Ji et al., 2023), however, it is not as easy to reproduce compared to running an empirical characterization tool like `ShOpt`.

As outlined in the state of the field, `ShOpt` is a tool built with the user experience in mind that attempts to bridge the strengths of existing PSF fitters. `ShOpt`'s combination of speed, user friendliness, and accuracy enable the science goals of the COSMOS-Web survey, detailed below.

The COMOS-Web survey is the largest cycle 1 JWST extragalactic survey according to area and prime time allocation (Casey, Kartaltepe, et al., 2023), and covers 0.54 $deg^2$ (Beichman et al., 2012; Rieke et al., 2023). Among other science goals, the COMOS-Web survey will use the JWST to detect thousands of galaxies in the Epoch of Reionization ($6 \sim z \sim 11$) to create one of the highest resolution large scale structure maps of the early universe (Casey, Kartaltepe, et al., 2023). JWST data has also been used to pick out active galactic nuclei from host galaxies (Zhuang & Shen, 2023) and indentify 15 candidate galaxies whose luminosities push the limits of our $\Lambda$CDM galaxy formation models (Casey, Akins, et al., 2023). These science cases all rely upon good PSF modeling and underscore the importance of `ShOpt`.

# Part VII

# The State of Julia for Scientific Machine Learning

## 36 Introduction — A Tale of Two Languages

Historically, Python has been the dominant language for machine learning research, including in the sciences. Python is intuitive and has a rich ecosystem across all of the natural sciences. Yet, the rise of Python as the go-to language for machine learning has been in many ways unnatural. Python is a scripting language, extremely slow, and challenging to maintain. The advent of Julia was, in many ways, designed to address the limitations of Python. Julia boasts a suite of language-level features that are designed for intuitive and fast numerical computations.

Since Julia's introduction in 2012, the language has experienced steady growth among practitioners Bezanson et al. (2017); jul; Kilpatrick (2024). Given Julia's specific intent to solve problems in scientific machine learning, it's natural to ask why it hasn't usurped Python in popularity within the natural sciences. Some of this can be attributed to the *momentum* Python has accumulated over the years within the community

Meyerovich & Rabkin (2013). However, with the rapid growth of scientific machine learning in recent years, it is to be seen whether Julia will see broader adoption.

In this work, we explore the readiness of Julia as the de-facto tool for scientific machine learning. We focus not only on the wealth of libraries at Julia's disposal but also its performance, design philosophy, and overall ergonomics. A key theme in this work is that different programming languages provide different abstractions that significantly change the way a user interacts with a scientific machine learning problem (SMLP), and Julia provides a very different set of abstractions than what is seen in other ecosystems. In contrast to previous perspectives Innes et al. (2018b); Gao et al. (2020), we argue that while the Julia ecosystem provides a number of useful abstractions for SMLPs, the limitations are severe enough to prevent it from widespread adoption. We conclude by calling for the community to discuss and address Julia's language-level limitations.

## 37 The Scientific Computing Ecosystem

In this Section, we compare Julia's ecosystems with its contemporaries. We compare primarily with Python as it is the go-to choice for most researchers tasked with a SMLP, however, we recognize that languages such as C, C++, Fortran, and Cuda see similar usage at smaller scales. We point out examples of SMLPs that Julia is especially useful for throughout.

**Linear Algebra** Arguably the most important part of the Julia ecosystem is the standard library for linear algebra. Fast numerical computations motivated Julia's inception. To achieve this, Julia employs a Just in Time (JIT) compiler to speed up its computation. The utility of the JIT compiler is seen across the language, but especially within its linear algebra library. For example, Julia is significantly faster than Python and about as fast as C in terms of random matrix multiplication and random matrix statistics Bezanson et al. (2017). Python matrix operations can be JIT compiled with Jax Bradbury et al. (2018), however, the exact performance gain compared to C or Julia has yet to be properly assessed. Julia also has a plethora of easy to use functions for matrix factorization and libraries for working with sparse graphs that can make matrix operations even faster The Julia Language (2024). This makes Julia the most compelling choice for SMLPs like SLAM Rosen et al. (2019b; 2020), where there is a natural graph sparsity to exploit. There are also some widely adopted libraries for sparsity in C and C++ Lourenco et al. (2022); Davis et al. (2020); Foster & Davis (2013); Kolodziej et al. (2019); Davis & Hu (2011); Davis (2005); Davis & Palamadai Natarajan (2010); Davis (2013); Davis & Duff (1997; 1999); Davis (2004a;b); Davis et al. (2004a;b); Amestoy et al. (1996; 2004); Rennich et al. (2016); Davis & Hager (1999; 2001; 2005; 2009); Chen et al. (2008); Yeralan et al. (2017); Davis (2011; 2006; 2019); Xu et al. (2022); Saad; Falgout et al. (2020).

**Constrained Optimization** In a similar flavor to numerical linear algebra, constrained optimization algorithms mesh well with Julia's strengths. Julia's constrained optimization algorithms are fast and mathematically sound, and in comparison to Python, much more abundant.

Julia has the most sophisticated support for specifying optimization problems on manifolds. Naive approaches to SMLPs do not account for inductive biases and physical constraints (e.g. Berman et al. (2024); Berman & McCleary (2024a); Rosen et al. (2020; 2019b); Absil et al. (2008b); Boumal (2023)). One approach to endowing physical constraints into a problem is by specifying a manifold that your data lives on. This is implemented in Julia with Manopt Bergmann (2022). While there are Python and Matlab ports of Manopt Boumal et al. (2014); Townsend et al. (2016), they offer only a strict subset of what the Julia library offers Bergmann (2022). The Python port in particular is very limited. There exist similar packages in R and C++ Huang et al. (2018); Martin et al. (2020) but they are not actively maintained.

Manopt is designed to constrain iterative updates to stay on a specified manifold. However, often times constraints are specified softly via loss functions in Physics Informed Neural Networks (PINNs). These PINNs are able to take into account boundary and initial conditions into account when solving for a physical law specified as a differential equation. Julia handles this wonderfully, with Flux.jl Innes (2018a); Innes et al. (2018a) and Lux.jl Pal (2023) both providing neural network backends that are compatible with the DiffEqFlux.jl Rackauckas et al. (2020) and NeuralPDE.jl Zubov et al. (2021) packages. Even beyond neural

methods, Julia has an extremely rich ecosystem for solving differential equations Rackauckas & Nie (2017b), however this is out of the scope of this paper. Lux.jl in particular supports arbitrary types, making it composible with arbitrary ODE/PDE solvers and even other languages (more on this in § 38). In contrast, there are no de facto solutions for physics informed machine learning in Python. That is, no existing solution accommodates neural networks specified in either TensorFlow Abadi et al. (2016), Haiku Hennigan et al. (2020), or PyTorch Paszke et al. (2019) simultaneously. As a result, we see standard architectures spread across different sub-ecosystems without any libraries available to bridge them together. For example, we see the canonical implementations for lagrangian neural networks Cranmer et al. (2020) and normalizing flows Kidger (2021); Halverson & Pandya (2024) implemented in Jax (possibly with Diffrax Kidger (2021)), and the canonical implemenations of Kolmogorov Arnold Networks Liu et al. (2024) and Deep Operator Networks Li et al. (2024); Lu et al. (2021) implemented in PyTorch.

Julia rounds out its suite of constrained optimization libraries with GeometricFlux.jl Tu (2020). GeometricFlux.jl again builds on the Flux.jl framework with specific layers and functionality for the $5(+1)G$ fields, which includes **G**raphs and sets, **G**rids and Euclidean Spaces, **G**roups and Homogenous Spaces, **G**eodesics and Manifolds, **G**auges and Bundles, and **G**eometric Algebra. As with PINNs, these same functionalities appear scattered throughout different Jax and Torch libraries that are often incompatible with one another. This support makes Julia an attractive choice for working on SMLPs such as excited state molecular dynamics Zou et al. (2023a); Shinn (2024). There is also the GraphNeuralNetworks.jl library Lucibello & other contributors (2021), which is more actively maintained than GeometricFlux.jl, and has similar features.

**Automatic Differentiation** Julia maintains several different packages for both forward mode and reverse mode automatic differentiation (AD) with ForwardDiff.jl **?** and Zygote.jl Innes et al. (2018a) respectively (see also Moses et al. (2021); Developers (2024a); Schäfer et al. (2021)). In other ecosystems, forward mode and reverse mode AD are typically handled within the same library, as is the case with PyTorch Paszke et al. (2019) and Jax Bradbury et al. (2018). In general, Julia's AD libraries are much more ambitious in scope, allowing for more overhead from the user at the potential cost of ease of use. We identify this as one of the few areas where there is a lot of friction when writing Julia code. One way in which this is realized is with custom gradient interventions and definitions. In Julia, macros for overriding and specifying differentiation rules are housed in many separate libraries, including Zygote.jl itself and the general purpose DiffRules.jl library Revels et al. (2018). Navigating the maze of different tools is often counter intuitive and involves much more domain knowledge than with other tools. In contrast, custom differentiation rules are very easy in Jax with the *@custom_JVP* and *@custom_VJP* macros.

Zygote.jl is most prominently used as the backend of the Flux.jl machine learning library. Additionally, the automatic differentiation tools described above also enable Optim.jl Mogensen & Riseth (2018) for use with SymbolicRegressions.jl. In the space of symbolic regression, Julia has a clear advantage in terms of support. The most widely adopted library for symbolic regression is SymbolicRegressions.jl Cranmer (2023). The Python alternative PySR is simply a wrapper for this library, which is in turn a dependency for PyTorch Language (2023).

**Probabilistic Programming** Julia boasts many Probabilistic Programming Languages (PPLs) built on top of it. These PPLs make SMLPs involving Bayesian statistics extremely simple to express, and include packages like SOSS.jl Scherrer & contributors (2024) and Turing.jl Ge et al. (2018). These PPLs also make domain specific languages (DSLs) like Dice Holtzen et al. (2020) unnecessary [17], as each of the PPLs is a superset of Julia. SOSS.jl uniquely provides a set of tools for working with measure theoretic objects not seen in other languages JuliaMath & contributors (2024); Scherrer & Schauer (2022); Scherrer & contributors (2024). Save for the measure theoretic support, a similar library also exists in Python with PyAutoFit Nightingale et al. (2021). Additionally, Jax Bradbury et al. (2018) with NumPyro Phan et al. (2019); Foreman-Mackey (2020) makes it exceptionally easy to specify probabilistic problems.

[17] Dice is extremely useful in theoretical programming language contexts, however.

## 38 Design Philosophy and Ergonomic Machine Learning

Julia was founded on strong design ideals; in this section we discuss Julia's features and how they enable fluid development and solving of SMLPs.

**Multiple Dispatch** As evidenced by Bezanson et al. (2017), Julia makes extensive use of multiple dispatch Muschevici et al. (2008), a method to automatically choose function behavior based on input types, compared to other languages. In practice, this can make writing code much more ergonomic for the end user. As an example, the Python/C++ library for computing correlation functions, TreeCorr Jarvis et al. (2004), has 18 different subclasses that the user must remember in order to correlate different kinds of objects. All of these subclasses are prefixed with single letters, such as KVCorrelation for Scalar-vector correlations and VVCorrelation for Vector-vector correlations. Needless to say, it can be difficult to keep track of all 18 combinations, but with Julia's multiple dispatch, we can safely define the same function 18 times for the different types of inputs without having to worry about specifying which one to call in practice. This is the approach also taken by CosmoCorr.jl Berman (2024). Other libraries that make extensive use of multiple dispatch include JuMP.jl Dunning et al. (2017), ForwardDiff.jl **?**, and even Julia's standard library.

**Composition vs. Inheritance** Julia's design philosophy and style guides revolve around the idea of highly composable interfaces. This naturally plays off of the use of multiple dispatch: functions are not attached to a single struct and instead are declared globally, potentially multiple times for different types. An example of this can be seen with Flux.jl Innes (2018a); Innes et al. (2018a). A neural network in Flux.jl is just a chain of functions that can act on different types. This makes Flux.jl much more ergonomic as it abstracts away the different input types you may provide to a neural network, eliminating the need to extend a module and format the input data in a certain way (i.e. as one would do with PyTorch Paszke et al. (2019)). It is for this reason that Flux.jl boasts that it is the library that *doesn't make you tensor*. The emphasis on composition is also useful for package management. Since there is very little shared state between different user defined structs due to multiple dispatch, packages are less likely to conflict. Even when there are conflicts, Julia's package manager Pkg.jl Kwong et al. (2023); Hanson (2024) is much easier to use. Python's package management on the contrary, which depends on Python virtual environment management with Pip or Conda, are notoriously error prone.

**Two Language Problem** The two language problem states that a programming language is either fast or easy to use, but not both. Thus, we often write performance critical code in C or C++ and call it from Python. This makes projects difficult to maintain as there are now multiple environments the user needs to keep track off. This also makes reproducibility and contribution more difficult as it requires everyone in the scientific community to know a second, low level technical language[18]. The most glaring example of this is PyTorch Paszke et al. (2019), which is a Python interface to a C++ backend. However, even libraries that provide much simpler functionality than a neural network have this issue. For example, TreeCorr also has a Python API for functions written in C++ Jarvis et al. (2004). Julia claims to solve the two language problem, allowing both flexible prototyping and deep performance optimization within the same language Bezanson et al. (2017). For example, the Flux.jl Innes (2018a); Innes et al. (2018a) and CosmoCorr.jl Berman (2024) libraries are able to efficiently train neural networks and estimate correlation functions all with high-level Julia code. In general, the interplay between prototyping and optimizing is a core design pattern when implementing solutions for SMLPs, and Julia's design uniquely facilitates this interplay.

## 39 Limitations of Julia for Scientific Machine Learning

In this section, we outline the limitations we find most severe in Julia for solving SMLPs.

**Lack of Software Engineering Features** In stark comparison with Python, Julia lacks many of the software engineering-oriented language and ecosystem features. Primarily, Julia has significantly less mature testing infrastructure than Python; the unittest and pytest Python libraries are much more robust than

[18] or, at least how to compile it with make or cmake, given the project doesn't provide a dockerfile or similar.

Julia's built in testing library. Additionally, Julia has virtually no support for more advanced testing methods such as property-based testing, symbolic execution, and contract-based testing, of which are universally employed by Python's large-scale numerical methods and machine learning libraries such as Pandas McKinney et al. (2011), NumPy Harris et al. (2020), SciPy Virtanen et al. (2020) , SymPy Meurer et al. (2017), Scikit-Learn Pedregosa et al. (2011), Jax Bradbury et al. (2018), Tinygrad Daniel & Contributors (2024), and Cupy Nishino & Loomis (2017) through the Hypothesis HypothesisWorks (2024), Crosshair Schanely (2024), and Deal life4 (2024) libraries. Julia also lacks an actively maintained static type checker Microsoft (2023); Shivarpatna Venkatesh et al. (2024); Developers (2024b). More rigorous testing methods are especially important in scientific computation, where precision and correctness of implementations of algorithms is paramount. In being able to deeply test properties of an algorithm, we can better reconcile epistemic (model) uncertainties Osband et al. (2023) from aleatoric (data) uncertainties.

**Complex Debugging Messages** On a language level, Julia's error messages have continually been an unaddressed pain point in the community. The complexity of multiple dispatch, as well as the notoriously long stack traces, make digesting Julia's error messages a tedious process. Additionally, a recent Julia Hub survey showed that Julia programmers cited complex debugging messages as one of the biggest technical hurdles for the language Hub (2024a;b).

**Limited Industry Adoption** While Julia may be gaining popularity in academic circles, machine learning research has a heavy industry presence unseen in many other disciplines. As such, open source contributions from industry is a huge factor in the overall adoption of a language or library. A key limitation of Julia is its lack of support from industry giants. Julia's biggest competitor is arguably Jax Bradbury et al. (2018), which is maintained by Google Deepmind — we see Jax as something that treats the symptoms of Python's problems, the most significant of which is its performance. Moreover, companies like HuggingFace have designed their API's for sharing model weights and datasets exclusively around the Python machine learning libraries, making it much more difficult for Julia users to share their scientific models with one another and necessitating the use of Python wrappers (e.g. Cheng (2024)). While Julia has a growing number of industry backers jul, there is a clear resource advantage for the Python ecosystems. Supporting this, the Julia Hub survey Hub (2024a;b) found 64% of users thought one of the biggest non-technical problems with the language was "there are not enough Julia users in my field or industry," a jump of $\approx 6\%$ from 2023. The survey also found that while 71% of respondents use Julia for research, only 16% reported "I use Julia in production for a business critical task" Hub (2024a;b). Industrial applications require robust testing and interoperability with existing tools, supporting the idea that the limitations laid out in this section are not at all unrelated.

**Poor Interoperability** Despite the annoyances of the two language problem laid out in § 38, users will inevitably need to use multiple languages until scientists start writing their domain specific applications in Julia. While calling Python, R, C, and C++ functions in Julia is relatively easy with the PythonCall.jl JuliaPy (2024), PyCall.jl Johnson & contributors (2024), RCall.jl Lai et al. (2024), and the Julia standard library function *ccall* Language (2024), the converse is often extremely difficult and unintuitive as you need to explicitly manage Julia's context and the passing of complex datatypes between languages. While JuliaPy (2024) handles the Julia to Python conversion most graciously, there is still much to be desired in calling Julia from other languages.

## 40 Conclusion and Call to Action

Not only does Julia match and in many ways exceed Python with its suite of general purpose scientific computing tools, but it also provides many more specific tools for solving SMLPs unseen in any other language. This is particularly true for constrained optimization problems that naturally arise in the natural sciences. Unfortunately, this has proven to be not enough for Julia to gain widespread adoption. While Julia has been slowly gaining traction, in comparison to the growth of scientific machine learning as a whole, Julia's growth has been quite slow.

Julia's lack of software engineering-centric features, lackluster debugging infrastructure, subpar industry adoption, its "rivalry" with Jax, and insufficient interopt all bottleneck further adoption. However, none of these limitations in isolation seem like dealbreakers. On paper, Julia *should* compete with Python. So again we ask, why doesn't it?

In the seminal paper, "Julia: A Fresh Approach to Numerical Computing" Bezanson et al. (2017), the initial vision for Julia is summarized in three points. 1) A programming language can be high-level, dynamic, and fast. 2) A programming language can be used for both scripting and deployment. 3) A programming language should supply a mechanism to easily abstract away unnecessary detail typically left to "the experts." Arguably, Julia today has addressed all three of these points and matured with respect to them. Yet, Julia remains relatively niche despite the growing potential user base.

In present year, Julia users have mostly moved away from improving the Julia language and are now more focused on developing libraries for specific projects. With love, we argue the Julia community has moved on too quickly. The state of machine learning is rapidly changing, and Julia has the potential to address many of the pain points in the community. However, if we do not address problems at the language level, we are in danger of repeating many of the mistakes of Python.

We believe the current state of Julia lacks vision. Julia needs a new constitution: a set of concrete goals for improvement, adoption, and outreach. While Python has a clear list of future goals pep, and individual ecosystems *within* Julia such as SciML have roadmaps Organization (2024), the Julia language itself has nothing but surface-level GitHub issues. It is only once we map out and solve these language-level issues that we can then ask if Julia is capable of succeeding Python as the de-facto language for scientific machine learning.

We ask the Julia and scientific machine learning communities: what is the future for Julia?

## References

Juliahub case studies. https://info.juliahub.com/case-studies. Accessed: 2024-08-31.

Python enhancement proposals. https://peps.python.org/. Accessed: 2024-09-03.

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for Large-Scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, Savannah, GA, November 2016. USENIX Association. ISBN 978-1-931971-33-1. URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

Timothy MC Abbott, Michel Aguena, Alex Alarcon, S Allam, O Alves, A Amon, F Andrade-Oliveira, James Annis, S Avila, D Bacon, et al. Dark energy survey year 3 results: Cosmological constraints from galaxy clustering and weak lensing. *Physical Review D*, 105(2):023520, 2022.

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024.

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008a. ISBN 978-0-691-13298-3.

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008b.

Paolo Aluffi. *Algebra: chapter 0*, volume 104. American Mathematical Soc., 2021.

Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996. doi: 10.1137/S0895479894278952. URL https://doi.org/10.1137/S0895479894278952.

Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. Algorithm 837: Amd, an approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):381–388, sep 2004. ISSN 0098-3500. doi: 10.1145/1024074.1024081. URL https://doi.org/10.1145/1024074.1024081.

Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *Advances in neural information processing systems*, 33:14927–14937, 2020.

Michael Artin. *Algebra*. Birkhäuser, 1998.

Michael Backenköhler, Paula Linh Kramer, Joschka Groß, Gerrit Großmann, Roman Joeres, Azat Tagirdzhanov, Dominique Sydow, Hamza Ibrahim, Floriane Odje, Verena Wolf, and Andrea Volkamer. TeachOpenCADD goes Deep Learning: Open-source Teaching Platform Exploring Molecular DL Applications. *ChemRxiv preprint*, 2023. doi: 10.26434/chemrxiv-2023-kz1pb.

Micaela B. Bagley, Steven L. Finkelstein, Sofía Rojas-Ruiz, James Diekmann, Keely D. Finkelstein, Mimi Song, Casey Papovich, Rachel S. Somerville, Ivano Baronchelli, and Y. Sophia Dai. Bright z∼9 Galaxies in Parallel: The Bright End of the Rest-UV Luminosity Function from HST Parallel Programs. *arXiv e-prints*, art. arXiv:2205.12980, May 2022. doi: 10.48550/arXiv.2205.12980.

Julia Balla, Siddharth Mishra-Sharma, Carolina Cuesta-Lazaro, Tommi Jaakkola, and Tess Smidt. A cosmic-scale benchmark for symmetry-preserving data processing. *arXiv preprint arXiv:2410.20516*, 2024.

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Eric Beh. Exploring how to simply approximate the p-value of a chi-squared statistic. *Austrian Journal of Statistics*, 47(3):63–75, 2018.

Charles A. Beichman, Marcia Rieke, Daniel Eisenstein, Thomas P. Greene, John Krist, Don McCarthy, Michael Meyer, and John Stansberry. Science opportunities with the near-IR camera (NIRCam) on the James Webb Space Telescope (JWST). In Mark C. Clampin, Giovanni G. Fazio, Howard A. MacEwen, and Jr. Oschmann, Jacobus M. (eds.), *Space Telescopes and Instrumentation 2012: Optical, Infrared, and Millimeter Wave*, volume 8442 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 84422N, September 2012. doi: 10.1117/12.925447.

Joel Bergé, Richard Massey, Quentin Baghi, and Pierre Touboul. Exponential shapelets: basis functions for data analysis of isolated features. *Monthly Notices of the Royal Astronomical Society*, 486(1):544–559, 2019.

Ronny Bergmann. Manopt.jl: Optimization on manifolds in Julia. *Journal of Open Source Software*, 7(70): 3866, 2022. doi: 10.21105/joss.03866.

Edward Berman. Cosmocorr: Cosmological correlation function estimator. https://github.com/EdwardBerman/CosmoCorr, 2024. Accessed: 2024-09-23.

Edward Berman and Jacob Ginesin. The state of julia for scientific machine learning. *arXiv preprint arXiv:2410.10908*, 2024.

Edward Berman and Jacqueline McCleary. Shopt.jl: A julia package for empirical point spread function characterization of jwst nircam data. *Journal of Open Source Software*, 9(100):6144, 2024a. doi: 10.21105/joss.06144. URL https://doi.org/10.21105/joss.06144.

Edward Berman and Jacqueline McCleary. Shopt.jl: A julia package for empirical point spread function characterization of jwst nircam data, 2024b.

Edward Berman and Jacqueline McCleary. On Differentiable Correlation Functions. In *American Astronomical Society Meeting Abstracts*, volume 245 of *American Astronomical Society Meeting Abstracts*, pp. 424.01, January 2025a.

Edward Berman and Jacqueline McCleary. On Differentiable Correlation Functions. In *American Astronomical Society Meeting Abstracts*, volume 245 of *American Astronomical Society Meeting Abstracts*, pp. 424.01, January 2025b.

Edward Berman, Sneh Pandya, Jacqueline McCleary, Marko Shuntov, Caitlin Casey, Nicole Drakos, Andreas Faisst, Steven Gillman, Ghassem Gozaliasl, Natalie Hogg, Jeyhan Kartaltepe, Anton Koekemoer, Wilfried Mercier, Diana Scognamiglio, COSMOS-Web, :, and The JWST Cosmic Origins Survey. On soft clustering for correlation estimators: Model uncertainty, differentiability, and surrogates, 2025. URL https://arxiv.org/abs/2504.06174.

Edward M. Berman, Jacqueline E. McCleary, Anton M. Koekemoer, Maximilien Franco, Nicole E. Drakos, Daizhong Liu, James W. Nightingale, Marko Shuntov, Diana Scognamiglio, Richard Massey, Guillaume Mahler, Henry Joy McCracken, Brant E. Robertson, Andreas L. Faisst, Caitlin M. Casey, Jeyhan S. Kartaltepe, and COSMOS-Web: The JWST Cosmic Origins Survey. Efficient point-spread function modeling with shopt.jl: A point-spread function benchmarking study with jwst nircam imaging. *The Astronomical Journal*, 168(4):174, sep 2024. doi: 10.3847/1538-3881/ad6a0f. URL https://dx.doi.org/10.3847/1538-3881/ad6a0f.

G. M. Bernstein and M. Jarvis. Shapes and shears, stars and smears: Optimal measurements for weak lensing. *The Astronomical Journal*, 123(2):583, feb 2002. doi: 10.1086/338085. URL https://dx.doi.org/10.1086/338085.

E. Bertin. Automated Morphometry with SExtractor and PSFEx. In I. N. Evans, A. Accomazzi, D. J. Mink, and A. H. Rots (eds.), *Astronomical Data Analysis Software and Systems XX*, volume 442 of *Astronomical Society of the Pacific Conference Series*, pp. 435, July 2011.

E. Bertin and S. Arnouts. SExtractor: Software for source extraction. **??***jnlA&AS*, 117:393–404, June 1996. doi: 10.1051/aas:1996164.

Emmanuel Bertin and Stephane Arnouts. Sextractor: Software for source extraction. *Astronomy and astrophysics supplement series*, 117(2):393–404, 1996.

Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences*, 10(2-3):191–203, 1984.

Simon Birrer, Anowar J. Shajib, Daniel Gilman, Aymeric Galan, Jelle Aalbers, Martin Millon, Robert Morgan, Giulia Pagano, Ji Won Park, Luca Teodori, Nicolas Tessore, Madison Ueland, Lyne Van de Vyvere, Sebastian Wagner-Carena, Ewoud Wempe, Lilan Yang, Xuheng Ding, Thomas Schmidt, Dominique Sluse, Ming Zhang, and Adam Amara. lenstronomy ii: A gravitational lensing software ecosystem. *Journal of Open Source Software*, 6(62):3283, 2021. doi: 10.21105/joss.03283. URL https://doi.org/10.21105/joss.03283.

Benjamin Bloem-Reddy, Yee Whye, et al. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21(90):1–61, 2020.

H Bonnet and Y Mellier. Statistical analysis of weak gravitational shear in the extended periphery of rich galaxy clusters. *Astronomy and Astrophysics, v. 303, p. 331*, 303:331, 1995.

Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023. doi: 10.1017/9781009166164. URL https://www.nicolasboumal.net/book.

Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, pp. 169–207, 2004.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J Bekkers, and Max Welling. Geometric and physical quantities improve e (3) equivariant message passing. *arXiv preprint arXiv:2110.02905*, 2021.

Johann Brehmer, Sönke Behrends, Pim de Haan, and Taco Cohen. Does equivariance matter at scale? *arXiv preprint arXiv:2410.23179*, 2024.

Rasmus Bro and Sijmen De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997. doi: https://doi.org/10.1002/(SICI)1099-128X(199709/10)11:5<393::AID-CEM483>3.0.CO;2-L. URL https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-128X%28199709/10%2911%3A5%3C393%3A%3AAID-CEM483%3E3.0.CO%3B2-L.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Corey Brummel-Smith, Greg Bryan, Iryna Butsky, Lauren Corlies, Andrew Emerick, John Forbes, Yusuke Fujimoto, Nathan J Goldbaum, Philipp Grete, Cameron B Hummels, et al. Enzo: An adaptive mesh refinement code for astrophysics (version 2.6). *Journal of Open Source Software*, 4(42):1636, 2019.

Greg L Bryan, Michael L Norman, Brian W O'Shea, Tom Abel, John H Wise, Matthew J Turk, Daniel R Reynolds, David C Collins, Peng Wang, Samuel W Skillman, et al. Enzo: An adaptive mesh refinement code for astrophysics. *The Astrophysical Journal Supplement Series*, 211(2):19, 2014.

Howard Bushouse, Jonathan Eisenhamer, Nadia Dencheva, James Davies, Perry Greenfield, Jane Morrison, Phil Hodge, Bernie Simon, David Grumm, Michael Droettboom, Edward Slavich, Megan Sosey, Tyler Pauly, Todd Miller, Robert Jedrzejewski, Warren Hack, David Davis, Steven Crawford, David Law, Karl Gordon, Michael Regan, Mihai Cara, Ken MacDonald, Larry Bradley, Clare Shanahan, William Jamieson, Mairan Teodoro, Thomas Williams, and Maria Pena-Guerrero. JWST Calibration Pipeline, 12 2023. URL https://github.com/spacetelescope/jwst.

Jonas Busk, Peter Bjørn Jørgensen, Arghya Bhowmik, Mikkel N Schmidt, Ole Winther, and Tejs Vegge. Calibrated uncertainty for molecular property prediction using ensembles of message passing neural networks. *Machine Learning: Science and Technology*, 3(1):015012, 2021.

Jean-Eric Campagne, François Lanusse, Joe Zuntz, Alexandre Boucaud, Santiago Casas, Minas Karamanis, David Kirkby, Denise Lanzieri, Yin Li, and Austin Peel. Jax-cosmo: An end-to-end differentiable and gpu accelerated cosmology library. *arXiv preprint arXiv:2302.05163*, 2023.

Sean M Carroll. *Spacetime and geometry*. Cambridge University Press, 2019.

Aarynn L Carter, Sasha Hinkley, Jens Kammerer, Andrew Skemer, Beth A Biller, Jarron M Leisenring, Maxwell A Millar-Blanchaer, Simon Petrus, Jordan M Stone, Kimberly Ward-Duong, et al. The jwst early release science program for direct observations of exoplanetary systems i: High-contrast imaging of the exoplanet hip 65426 b from 2 to 16 $\mu$m. *The Astrophysical journal letters*, 951(1):L20, 2023.

Caitlin M. Casey, Jeyhan S. Kartaltepe, Nicole E. Drakos, Maximilien Franco, Santosh Harish, Louise Paquereau, Olivier Ilbert, Caitlin Rose, Isabella G. Cox, James W. Nightingale, Brant E. Robertson, John D. Silverman, Anton M. Koekemoer, Richard Massey, Henry Joy McCracken, Jason Rhodes, Hollis B. Akins, Aristeidis Amvrosiadis, Rafael C. Arango-Toro, Micaela B. Bagley, Angela Bongiorno, Peter L. Capak, Jaclyn B. Champagne, Nima Chartab, Oscar A. Chavez Ortiz, Katherine Chworowsky, Kevin C. Cooke, Olivia R. Cooper, Behnam Darvish, Xuheng Ding, Andreas L. Faisst, Steven L. Finkelstein, Seiji

Fujimoto, Fabrizio Gentile, Steven Gillman, Katriona M. L. Gould, Ghassem Gozaliasl, Christopher C. Hayward, Qiuhan He, Shoubaneh Hemmati, Michaela Hirschmann, Knud Jahnke, Shuowen Jin, Ali Ahmad Khostovan, Vasily Kokorev, Erini Lambrides, Clotilde Laigle, Rebecca L. Larson, Gene C. K. Leung, Daizhong Liu, Tobias Liaudat, Arianna S. Long, Georgios Magdis, Guillaume Mahler, Vincenzo Mainieri, Sinclaire M. Manning, Claudia Maraston, Crystal L. Martin, Jacqueline E. McCleary, Jed McKinney, Conor J. R. McPartland, Bahram Mobasher, Rohan Pattnaik, Alvio Renzini, R. Michael Rich, David B. Sanders, Zahra Sattari, Diana Scognamiglio, Nick Scoville, Kartik Sheth, Marko Shuntov, Martin Sparre, Tomoko L. Suzuki, Margherita Talia, Sune Toft, Benny Trakhtenbrot, C. Megan Urry, Francesco Valentino, Brittany N. Vanderhoof, Eleni Vardoulaki, John R. Weaver, Katherine E. Whitaker, Stephen M. Wilkins, Lilan Yang, and Jorge A. Zavala. Cosmos-web: An overview of the jwst cosmic origins survey, 2023a.

Caitlin M Casey, Jeyhan S Kartaltepe, Nicole E Drakos, Maximilien Franco, Santosh Harish, Louise Paquereau, Olivier Ilbert, Caitlin Rose, Isabella G Cox, James W Nightingale, et al. Cosmos-web: an overview of the jwst cosmic origins survey. *The Astrophysical Journal*, 954(1):31, 2023b.

Federico Cassano, John Gouwar, Francesca Lucchetti, Claire Schlesinger, Carolyn Jane Anderson, Michael Greenberg, Abhinav Jangda, and Arjun Guha. Knowledge transfer from high-resource to low-resource programming languages for code llms, 2023.

Augusto T. Chantada, Susana J. Landau, Pavlos Protopapas, Claudia G. Scóccola, and Cecilia Garraffo. Cosmology-informed neural networks to solve the background dynamics of the universe. *Phys. Rev. D*, 107:063523, Mar 2023. doi: 10.1103/PhysRevD.107.063523. URL https://link.aps.org/doi/10.1103/PhysRevD.107.063523.

Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3), oct 2008. ISSN 0098-3500. doi: 10.1145/1391989.1391995. URL https://doi.org/10.1145/1391989.1391995.

Ching Wen Cheng. Transformers.jl: Hugging face models, 2024. URL https://chengchingwen.github.io/Transformers.jl/stable/huggingface/. Accessed: 2024-09-06.

Mostafa Cherif, Tobías I Liaudat, Jonathan Kern, Christophe Kervazo, and Jérôme Bobin. Uncertainty quantification for fast reconstruction methods using augmented equivariant bootstrap: Application to radio interferometry. *arXiv preprint arXiv:2410.23178*, 2024.

Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. Nvidia a100 tensor core gpu: Performance and innovation. *IEEE Micro*, 41(2):29–35, 2021.

Coblentz Society. Sulfur dioxide ($so_2$) infrared spectrum. https://webbook.nist.gov/cgi/cbook.cgi?ID=C7446095&Index=0&Type=IR-SPEC, 1977. URL https://webbook.nist.gov/cgi/cbook.cgi?ID=C7446095&Index=0&Type=IR-SPEC. Spectrum recorded by Dow Chemical Company, digitized by NIST.

Dark Energy Survey Collaboration:, T Abbott, FB Abdalla, J Aleksić, S Allam, A Amara, D Bacon, E Balbinot, M Banerji, K Bechtol, et al. The dark energy survey: more than dark energy–an overview. *Monthly Notices of the Royal Astronomical Society*, 460(2):1270–1299, 2016.

Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

Carolina Cuesta-Lazaro and Siddharth Mishra-Sharma. Point cloud approach to generative modeling for galaxy surveys at the field level. *Physical Review D*, 109(12):123531, 2024.

George Hotz Daniel and Contributors. tinygrad: Simple and small neural network library in python. https://github.com/tinygrad/tinygrad, 2024. GitHub repository.

Dark Energy Survey Collaboration. DES Y3 PIFF Data Release. https://des.ncsa.illinois.edu/releases/y3a2/Y3piff, 2024. Accessed: 2024-12-21.

T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, 2006. doi: 10.1137/1.9780898718881. URL https://doi.org/10.1137/1.9780898718881.

Timothy A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):165–195, jun 2004a. ISSN 0098-3500. doi: 10.1145/992200.992205. URL https://doi.org/10.1145/992200.992205.

Timothy A. Davis. Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):196–199, jun 2004b. ISSN 0098-3500. doi: 10.1145/992200.992206. URL https://doi.org/10.1145/992200.992206.

Timothy A. Davis. Algorithm 849: A concise sparse cholesky factorization package. *ACM Trans. Math. Softw.*, 31(4):587–591, dec 2005. ISSN 0098-3500. doi: 10.1145/1114268.1114277. URL https://doi.org/10.1145/1114268.1114277.

Timothy A. Davis. Algorithm 915, suitesparseqr: Multifrontal multithreaded rank-revealing sparse qr factorization. *ACM Trans. Math. Softw.*, 38(1), dec 2011. ISSN 0098-3500. doi: 10.1145/2049662.2049670. URL https://doi.org/10.1145/2049662.2049670.

Timothy A. Davis. Algorithm 930: Factorize: An object-oriented linear system solver for matlab. *ACM Trans. Math. Softw.*, 39(4), jul 2013. ISSN 0098-3500. doi: 10.1145/2491491.2491498. URL https://doi.org/10.1145/2491491.2491498.

Timothy A. Davis. Algorithm 1000: Suitesparse:graphblas: Graph algorithms in the language of sparse linear algebra. *ACM Trans. Math. Softw.*, 45(4), dec 2019. ISSN 0098-3500. doi: 10.1145/3322125. URL https://doi.org/10.1145/3322125.

Timothy A. Davis and Iain S. Duff. An unsymmetric-pattern multifrontal method for sparse lu factorization. *SIAM Journal on Matrix Analysis and Applications*, 18(1):140–158, 1997. doi: 10.1137/S0895479894246905. URL https://doi.org/10.1137/S0895479894246905.

Timothy A. Davis and Iain S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Trans. Math. Softw.*, 25(1):1–20, mar 1999. ISSN 0098-3500. doi: 10.1145/305658.287640. URL https://doi.org/10.1145/305658.287640.

Timothy A. Davis and William W. Hager. Modifying a sparse cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 20(3):606–627, 1999. doi: 10.1137/S0895479897321076. URL https://doi.org/10.1137/S0895479897321076.

Timothy A. Davis and William W. Hager. Multiple-rank modifications of a sparse cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 22(4):997–1013, 2001. doi: 10.1137/S0895479899357346. URL https://doi.org/10.1137/S0895479899357346.

Timothy A. Davis and William W. Hager. Row modifications of a sparse cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 26(3):621–639, 2005. doi: 10.1137/S089547980343641X. URL https://doi.org/10.1137/S089547980343641X.

Timothy A. Davis and William W. Hager. Dynamic supernodes in sparse cholesky update/downdate and triangular solves. *ACM Trans. Math. Softw.*, 35(4), feb 2009. ISSN 0098-3500. doi: 10.1145/1462173.1462176. URL https://doi.org/10.1145/1462173.1462176.

Timothy A. Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1), dec 2011. ISSN 0098-3500. doi: 10.1145/2049662.2049663. URL https://doi.org/10.1145/2049662.2049663.

Timothy A. Davis and Ekanathan Palamadai Natarajan. Algorithm 907: Klu, a direct sparse solver for circuit simulation problems. *ACM Trans. Math. Softw.*, 37(3), sep 2010. ISSN 0098-3500. doi: 10.1145/1824801.1824814. URL https://doi.org/10.1145/1824801.1824814.

Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):353–376, sep 2004a. ISSN 0098-3500. doi: 10.1145/1024074.1024079. URL https://doi.org/10.1145/1024074.1024079.

Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm. *ACM Trans. Math. Softw.*, 30(3):377–380, sep 2004b. ISSN 0098-3500. doi: 10.1145/1024074.1024080. URL https://doi.org/10.1145/1024074.1024080.

Timothy A. Davis, William W. Hager, Scott P. Kolodziej, and S. Nuri Yeralan. Algorithm 1003: Mongoose, a graph coarsening and partitioning library. *ACM Trans. Math. Softw.*, 46(1), mar 2020. ISSN 0098-3500. doi: 10.1145/3337792. URL https://doi.org/10.1145/3337792.

ChainRules Core Developers. Chainrules.jl. https://github.com/JuliaDiff/ChainRules.jl, 2024a. URL https://github.com/JuliaDiff/ChainRules.jl. GitHub repository.

Mypy Developers. Mypy: Optional static typing for python. https://github.com/python/mypy, 2024b. Accessed: 2024-09-07.

Ilias Diakonikolas, Samuel B Hopkins, Daniel Kane, and Sushrut Karmalkar. Robustly learning any clusterable mixture of gaussians. *arXiv preprint arXiv:2005.06417*, 2020.

Keith R Dienes, Lucien Heurtier, Fei Huang, Doojin Kim, Tim MP Tait, and Brooks Thomas. Stasis in an expanding universe: A recipe for stable mixed-component cosmological eras. *Physical Review D*, 105(2): 023530, 2022.

Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM review*, 59(2):295–320, 2017.

Carlos Esteves. Theoretical aspects of group equivariant neural networks. *arXiv preprint arXiv:2004.05154*, 2020.

R Falgout et al. Hypre: Scalable linear solvers and multigrid methods, 2020.

Brandon Y Feng, Rodrigo Ferrer-Chávez, Aviad Levis, Jason J Wang, Katherine L Bouman, and William T Freeman. Exoplanet detection via differentiable rendering. *arXiv preprint arXiv:2501.01912*, 2025.

Steven L. Finkelstein, Micaela B. Bagley, Pablo Arrabal Haro, Mark Dickinson, Henry C. Ferguson, Jeyhan S. Kartaltepe, Casey Papovich, Denis Burgarella, Dale D. Kocevski, Marc Huertas-Company, Kartheik G. Iyer, Anton M. Koekemoer, Rebecca L. Larson, Pablo G. Pérez-González, Caitlin Rose, Sandro Tacchella, Stephen M. Wilkins, Katherine Chworowsky, Aubrey Medrano, Alexa M. Morales, Rachel S. Somerville, L. Y. Aaron Yung, Adriano Fontana, Mauro Giavalisco, Andrea Grazian, Norman A. Grogin, Lisa J. Kewley, Allison Kirkpatrick, Peter Kurczynski, Jennifer M. Lotz, Laura Pentericci, Nor Pirzkal, Swara Ravindranath, Russell E. Ryan, Jonathan R. Trump, Guang Yang, Omar Almaini, Ricardo O. Amorín, Marianna Annunziatella, Bren E. Backhaus, Guillermo Barro, Peter Behroozi, Eric F. Bell, Rachana Bhatawdekar, Laura Bisigello, Volker Bromm, Véronique Buat, Fernando Buitrago, Antonello Calabrò, Caitlin M. Casey, Marco Castellano, Óscar A. Chávez Ortiz, Laure Ciesla, Nikko J. Cleri, Seth H. Cohen, Justin W. Cole, Kevin C. Cooke, M. C. Cooper, Asantha R. Cooray, Luca Costantin, Isabella G. Cox, Darren Croton, Emanuele Daddi, Romeel Davé, Alexander de La Vega, Avishai Dekel, David Elbaz, Vicente Estrada-Carpenter, Sandra M. Faber, Vital Fernández, Keely D. Finkelstein, Jonathan Freundlich, Seiji Fujimoto, Ángela García-Argumánez, Jonathan P. Gardner, Eric Gawiser, Carlos Gómez-Guijarro, Yuchen Guo, Kurt Hamblin, Timothy S. Hamilton, Nimish P. Hathi, Benne W. Holwerda, Michaela Hirschmann, Taylor A. Hutchison, Anne E. Jaskot, Saurabh W. Jha, Shardha Jogee, Stéphanie Juneau, Intae Jung, Susan A. Kassin, Aurélien Le Bail, Gene C. K. Leung, Ray A. Lucas, Benjamin Magnelli,

Kameswara Bharadwaj Mantha, Jasleen Matharu, Elizabeth J. McGrath, Daniel H. McIntosh, Emiliano Merlin, Bahram Mobasher, Jeffrey A. Newman, David C. Nicholls, Viraj Pandya, Marc Rafelski, Kaila Ronayne, Paola Santini, Lise-Marie Seillé, Ekta A. Shah, Lu Shen, Raymond C. Simons, Gregory F. Snyder, Elizabeth R. Stanway, Amber N. Straughn, Harry I. Teplitz, Brittany N. Vanderhoof, Jesús Vega-Ferrero, Weichen Wang, Benjamin J. Weiner, Christopher N. A. Willmer, Stijn Wuyts, Jorge A. Zavala, and Ceers Team. A Long Time Ago in a Galaxy Far, Far Away: A Candidate z $\sim$ 12 Galaxy in Early JWST CEERS Imaging. **??***jnlApJL*, 940(2):L55, December 2022. doi: 10.3847/2041-8213/ac966e.

Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pp. 3165–3176. PMLR, 2020.

Daniel Foreman-Mackey. An introduction to numpyro. https://dfm.io/posts/intro-to-numpyro/, 2020. Accessed: 2024-08-30.

Leslie V. Foster and Timothy A. Davis. Algorithm 933: Reliable calculation of numerical rank, null space bases, pseudoinverse solutions, and basic solutions using suitesparseqr. *ACM Trans. Math. Softw.*, 40(1), oct 2013. ISSN 0098-3500. doi: 10.1145/2513109.2513116. URL https://doi.org/10.1145/2513109.2513116.

Wendy L Freedman, Barry F Madore, In Sung Jang, Taylor J Hoyt, Abigail J Lee, and Kayla A Owens. Status report on the chicago-carnegie hubble program (cchp): three independent astrophysical determinations of the hubble constant using the james webb space telescope. *arXiv preprint arXiv:2408.06153*, 2024.

Jiahui Fu, Yilun Du, Kurran Singh, Joshua B Tenenbaum, and John J Leonard. Neuse: Neural se (3)-equivariant embedding for consistent spatial understanding with objects. *arXiv preprint arXiv:2303.07308*, 2023.

Shenming Fu, Ian Dell'Antonio, Ranga-Ram Chary, Douglas Clowe, M. C. Cooper, Megan Donahue, August Evrard, Mark Lacy, Tod Lauer, Binyang Liu, Jacqueline McCleary, Massimo Meneghetti, Hironao Miyatake, Mireia Montes, Priyamvada Natarajan, Michelle Ntampaka, Elena Pierpaoli, Marc Postman, Jubee Sohn, Keiichi Umetsu, Yousuke Utsumi, and Gillian Wilson. Lovoccs. i. survey introduction, data processing pipeline, and early science results. *The Astrophysical Journal*, 933(1):84, jul 2022. doi: 10.3847/1538-4357/ac68e8. URL https://dx.doi.org/10.3847/1538-4357/ac68e8.

Dominik Fuchsgruber, Tom Wollschläger, and Stephan Günnemann. Energy-based epistemic uncertainty for graph neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=6vNPPtWH1Q.

Kaifeng Gao, Gang Mei, Francesco Piccialli, Salvatore Cuomo, Jingzhi Tu, and Zenan Huo. Julia language in machine learning: Algorithms, applications, and open issues. *Computer Science Review*, 37:100254, 2020.

Cecilia Garraffo. Astroai. *Bulletin of the AAS*, 56(2), feb 7 2024. https://baas.aas.org/pub/2024n2i120p05.

Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A language for flexible probabilistic inference. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1682–1690. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/ge18b.html.

Roy C Geary. The ratio of the mean deviation to the standard deviation as a test of normality. *Biometrika*, 27(3/4):310–332, 1935.

Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks, 2022. URL https://arxiv.org/abs/2207.09453.

Mario Geiger, Tess Smidt, Alby M., Benjamin Kurt Miller, Wouter Boomsma, Bradley Dice, Kostiantyn Lapchevskyi, Maurice Weiler, Michał Tyszkiewicz, Simon Batzner, Dylan Madisetti, Martin Uhrin, Jes Frellsen, Nuri Jung, Sophia Sanborn, Mingjian Wen, Josh Rackers, Marcel Rød, and Michael Bailey. Euclidean neural networks: e3nn, apr 2022. URL https://doi.org/10.5281/zenodo.6459381.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

Jacob Golomb, Graça Rocha, Tiffany Meshkat, Michael Bottom, Dimitri Mawet, Bertrand Mennesson, Gautam Vasisht, and Jason Wang. Planetevidence: Planet or noise? *The Astronomical Journal*, 162(6): 304, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 6.2. 2.3 softmax units for multinoulli output distributions. *Deep learning*, 180, 2016.

Nate Gruver, Marc Anton Finzi, Micah Goldblum, and Andrew Gordon Wilson. The lie derivative for measuring learned equivariance. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JL7Va5Vy15J.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Brian C Hall. *Lie groups, Lie algebras, and representations*. Springer, 2013.

James Halverson and Sneh Pandya. Generality and persistence of cosmological stasis. *Phys. Rev. D*, 110: 075041, Oct 2024. doi: 10.1103/PhysRevD.110.075041. URL https://link.aps.org/doi/10.1103/PhysRevD.110.075041.

Eric P. Hanson. Julia's package registration tooling, 2024. URL https://ericphanson.com/blog/2024/julias-package-registration-tooling/. Accessed: 2024-10-28. https://ericphanson.com/blog/2024/julias-package-registration-tooling/.

Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Andrew P. Hearin, Duncan Campbell, Erik Tollerud, Peter Behroozi, Benedikt Diemer, Nathan J. Goldbaum, Elise Jennings, Alexie Leauthaud, Yao-Yuan Mao, Surhud More, John Parejko, Manodeep Sinha, Brigitta Sipöcz, and Andrew Zentner. Forward modeling of large-scale structure: An open-source approach with halotools. *The Astronomical Journal*, 154(5):190, oct 2017. doi: 10.3847/1538-3881/aa859f. URL https://dx.doi.org/10.3847/1538-3881/aa859f.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2024. URL http://github.com/google/flax.

Tom Hennigan, Trevor Cai, Tamara Norman, Lena Martens, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

Bryan Hilbert, Johannes Sahlmann, Kevin Volk, Shannon Osborne, Dthatte, Marshall Perrin, Lauren Chambers, Edward Slavich, Jo Taylor, Erik Tollerud, and P. L. Lim. spacetelescope/mirage: First github release. Zenodo, October 2019.

Christopher Hirata and Uroš Seljak. Shear calibration biases in weak-lensing surveys. *Monthly Notices of the Royal Astronomical Society*, 343(2):459–480, 2003.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pp. 409–426, 1994.

Matthew D Hoffman, Andrew Gelman, et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

Steven Holtzen, Guy Van den Broeck, and Todd Millstein. Scaling exact inference for discrete probabilistic programs. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):1–31, 2020.

Haojie Huang, Dian Wang, Xupeng Zhu, Robin Walters, and Robert Platt. Edge grasp network: A graph-based se (3)-invariant approach to grasp detection. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3882–3888. IEEE, 2023.

Haojie Huang, Owen Howell, Dian Wang, Xupeng Zhu, Robin Walters, and Robert Platt. Fourier transporter: Bi-equivariant robotic manipulation in 3d. *arXiv preprint arXiv:2401.12046*, 2024a.

Haojie Huang, Dian Wang, Arsh Tangri, Robin Walters, and Robert Platt. Leveraging symmetries in pick and place. *The International Journal of Robotics Research*, 43(4):550–571, 2024b.

Wen Huang, P-A Absil, Kyle A Gallivan, and Paul Hand. Roptlib: an object-oriented c++ library for optimization on riemannian manifolds. *ACM Transactions on Mathematical Software (TOMS)*, 44(4): 1–21, 2018.

Julia Hub. Juliacon 2024 | function room | day 3, 2024a. URL https://www.youtube.com/watch?v=qru5G5Yp77E&t=11635s. Accessed: 2024-10-24, timestamp 3:13:40.

Julia Hub. Sample form, 2024b. URL https://form.jotform.com/241373057274355. Accessed: 2024-10-24. form.jotform.com/241373057274355.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506, 2021.

Frederik Boe Hüttel, Filipe Rodrigues, and Francisco Câmara Pereira. Deep evidential learning for bayesian quantile regression. *arXiv preprint arXiv:2308.10650*, 2023.

HypothesisWorks. Hypothesis: A python library for property-based testing. https://github.com/HypothesisWorks/hypothesis, 2024. Accessed: 2024-09-07.

Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah. Fashionable modelling with flux. *arXiv preprint arXiv:1811.01457*, 2018a.

Mike Innes. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 3(25):602, 2018a.

Mike Innes. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 2018b. doi: 10.21105/joss.00602.

Mike Innes, Stefan Karpinski, Viral Shah, David Barber, PLEPS Saito Stenetorp, Tim Besard, James Bradbury, Valentin Churavy, Simon Danisch, Alan Edelman, et al. On machine learning and programming languages. Association for Computing Machinery (ACM), 2018b.

Mike Innes, Alan Edelman, Keno Fischer, Chris Rackauckas, Elliot Saba, Viral B Shah, and Will Tebbutt. A differentiable programming system to bridge machine learning and scientific computing. *arXiv preprint arXiv:1907.07587*, 2019.

Yesukhei Jagvaral, Francois Lanusse, and Rachel Mandelbaum. Geometric deep learning for galaxy-halo connection: a case study for galaxy intrinsic alignments. *arXiv preprint arXiv:2409.18761*, 2024.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

M Jarvis, G M Bernstein, A Amon, C Davis, P F Lé get, K Bechtol, I Harrison, M Gatti, A Roodman, C Chang, R Chen, A Choi, S Desai, A Drlica-Wagner, D Gruen, R A Gruendl, A Hernandez, N MacCrann, J Meyers, A Navarro-Alsina, S Pandey, A A Plazas, L F Secco, E Sheldon, M A Troxel, S Vorperian, K Wei, J Zuntz, T M C Abbott, M Aguena, S Allam, S Avila, S Bhargava, S L Bridle, D Brooks, A Carnero Rosell, M Carrasco Kind, J Carretero, M Costanzi, L N da Costa, J De Vicente, H T Diehl, P Doel,

S Everett, B Flaugher, P Fosalba, J Frieman, J García-Bellido, E Gaztanaga, D W Gerdes, G Gutierrez, S R Hinton, D L Hollowood, K Honscheid, D J James, S Kent, K Kuehn, N Kuropatkin, O Lahav, M A G Maia, M March, J L Marshall, P Melchior, F Menanteau, R Miquel, R L C Ogando, F Paz-Chinchón, E S Rykoff, E Sanchez, V Scarpine, M Schubnell, S Serrano, I Sevilla-Noarbe, M Smith, E Suchyta, M E C Swanson, G Tarle, T N Varga, A R Walker, W Wester, and R D Wilkinson and. Dark energy survey year 3 results: point spread function modelling. *Monthly Notices of the Royal Astronomical Society*, 501(1):1282–1299, nov 2020. doi: 10.1093/mnras/staa3679. URL https://doi.org/10.1093%2Fmnras%2Fstaa3679.

Mike Jarvis and collaborators. Treecorr: A python package for fast two-point correlation functions. https://github.com/rmjarvis/TreeCorr, 2024. Accessed: 2024-12-07.

Mike Jarvis, Gary Bernstein, and Bhuvnesh Jain. The skewness of the aperture mass statistic. *Monthly Notices of the Royal Astronomical Society*, 352(1):338–352, 2004.

Mike Jarvis, GM Bernstein, A Amon, C Davis, PF Léget, K Bechtol, I Harrison, M Gatti, A Roodman, C Chang, et al. Dark energy survey year 3 results: point spread function modelling. *Monthly Notices of the Royal Astronomical Society*, 501(1):1282–1299, 2021.

Edwin T Jaynes. Prior probabilities. *IEEE Transactions on systems science and cybernetics*, 4(3):227–241, 1968.

Zhiyuan Ji, Christina C Williams, Sandro Tacchella, Katherine A Suess, William M Baker, Stacey Alberts, Andrew J Bunker, Benjamin D Johnson, Brant Robertson, Fengwu Sun, et al. Jades+ jems: A detailed look at the buildup of central stellar cores and suppression of star formation in galaxies at redshifts 3< z< 4.5. *arXiv preprint arXiv:2305.18518*, 2023.

Mingxi Jia, Dian Wang, Guanang Su, David Klee, Xupeng Zhu, Robin Walters, and Robert Platt. Seil: Simulation-augmented equivariant imitation learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1845–1851. IEEE, 2023.

Steven G. Johnson and contributors. Pycall.jl: Calling python functions from the julia language. https://github.com/JuliaPy/PyCall.jl, 2024. Accessed: 2024-08-30.

JuliaMath and contributors. Measuretheory.jl, 2024. URL https://github.com/JuliaMath/MeasureTheory.jl. GitHub repository.

JuliaPy. Pythoncall.jl. https://github.com/JuliaPy/PythonCall.jl, 2024. Accessed: 2024-10-22.

Mira Jürgens, Nis Meinert, Viktor Bengs, Eyke Hüllermeier, and Willem Waegeman. Is epistemic uncertainty faithfully represented by evidential deep learning methods? *arXiv preprint arXiv:2402.09056*, 2024.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: scalable deep reinforcement learning for vision-based robotic manipulation. corr abs/1806.10293 (2018). *arXiv preprint arXiv:1806.10293*, 2018.

Akanksha Kapoor and Abhishek Singhal. A comparative study of k-means, k-means++ and fuzzy c-means clustering algorithms. In *2017 3rd international conference on computational intelligence & communication technology (CICT)*, pp. 1–6. IEEE, 2017.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in neural information processing systems*, 34:852–863, 2021.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

Martin Kilbinger. Cosmology with cosmic shear observations: a review. *Reports on Progress in Physics*, 78 (8):086901, 2015.

Logan Kilpatrick. Some julia growth & usage stats, 2024. URL https://discourse.julialang.org/t/some-julia-growth-usage-stats/112547. Accessed: 2024-09-02.

David Klee, Jung Yeon Park, Robert Platt, and Robin Walters. A comparison of equivariant vision models with imagenet pre-training. In *NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations*, 2023.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

Vasily I. Kokorev, Georgios E. Magdis, Iary Davidzon, Gabriel Brammer, Francesco Valentino, Emanuele Daddi, Laure Ciesla, Daizhong Liu, Shuowen Jin, Isabella Cortzen, Ivan Delvecchio, Clara Giménez-Arteaga, Carlos Gómez-Guijarro, Mark Sargent, Sune Toft, and John R. Weaver. The Evolving Interstellar Medium of Star-forming Galaxies, as Traced by Stardust. **??***jnlApJ*, 921(1):40, November 2021. doi: 10.3847/1538-4357/ac18ce.

Scott P. Kolodziej, Mohsen Aznaveh, Matthew Bullock, Jarrett David, Timothy A. Davis, Matthew Henderson, Yifan Hu, and Read Sandstrom. The suitesparse matrix collection website interface. *Journal of Open Source Software*, 4(35):1244, 2019. doi: 10.21105/joss.01244. URL https://doi.org/10.21105/joss.01244.

Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch-gordan nets: a fully fourier space spherical convolutional neural network, 2018. URL https://arxiv.org/abs/1806.09231.

Igor Kononenko. Bayesian neural networks. *Biological Cybernetics*, 61(5):361–370, 1989.

Andrey V Kravtsov, Anatoly A Klypin, and Alexei M Khokhlov. Adaptive refinement tree: a new high-resolution n-body code for cosmological simulations. *The Astrophysical Journal Supplement Series*, 111 (1):73, 1997.

John E Krist, Richard N Hook, and Felix Stoehr. 20 years of hubble space telescope optical modeling using tiny tim. In *Optical Modeling and Performance Predictions V*, volume 8127, pp. 166–181. SPIE, 2011.

H.T. Kung, Vikas Natesh, and Andrew Sabot. Cake: Matrix multiplication using constant-bandwidth blocks. In *SC21: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–13, 2021. doi: 10.1145/3458817.3476166.

Tom Kwong, Kristoffer Carlsson, Stefan Karpinski, Elliot Saba, et al. Pkg.jl: Julia's package manager. https://github.com/JuliaLang/Pkg.jl, 2023. Accessed: 2024-08-30.

Randy Lai, Simon Byrne, Douglas Bates, Phillip Alday, Viral B. Shah, Milan Bouchet-Valat, Dave Kleinschmidt, Dilum Aluthge, Changcheng Li, Ranjan Anantharaman, Antoine Baldassari, Steven G. Johnson, lbilli, Robert Feldt, Páll Haraldsson, Pietro Monticone, Michael Hatherly, Lukas Elmiger, Iain Dunning, and Rik Huijzer. JuliaInterop/RCall.jl: v0.14.4, August 2024. URL https://doi.org/10.5281/zenodo.12774049.

Claire Lamman, Eleni Tsaprazi, Jingjing Shi, Nikolina Niko Šarčević, Susan Pyne, Elisa Legnani, and Tassia Ferreira. The ia guide: A breakdown of intrinsic alignment formalisms. *arXiv preprint arXiv:2309.08605*, 2023.

Stephen D Landy and Alexander S Szalay. Bias and variance of angular correlation functions. *Astrophysical Journal, Part 1 (ISSN 0004-637X), vol. 412, no. 1, p. 64-71.*, 412:64–71, 1993.

The Julia Language. Calling c and fortran code, 2024. URL https://github.com/JuliaLang/julia/blob/master/doc/src/manual/calling-c-and-fortran-code.md. Accessed: 2024-09-06.

The Julia Programming Language. Interpretable machine learning with symbolicregression.jl | miles cranmer | juliacon 2023, 2023. URL https://www.youtube.com/watch?v=QLiQJDNwt4o. Accessed: 2024-10-13.

François Lanusse et al. The dawes review 10: The impact of deep learning for the analysis of galaxy surveys. *Publications of the Astronomical Society of Australia*, 40:e001, 2023.

Denise Lanzieri, François Lanusse, Chirag Modi, Benjamin Horowitz, Joachim Harnois-Déraps, and Jean-Luc Starck. Forecasting the power of higher order weak-lensing statistics with automatically differentiable simulations. *Astronomy & Astrophysics*, 679:A61, 2023.

Hannah Lawrence. Barron's theorem for equivariant networks. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022.

Pablo Lemos, Adam Coogan, Yashar Hezaveh, and Laurence Perreault-Levasseur. Sampling-based accuracy testing of posterior estimators for general inference. In *International Conference on Machine Learning*, pp. 19256–19273. PMLR, 2023.

Dan Levi, Liran Gispan, Niv Giladi, and Ethan Fetaya. Evaluating and calibrating uncertainty prediction in regression tasks. *Sensors*, 22(15):5540, 2022a.

Dan Levi, Liran Gispan, Niv Giladi, and Ethan Fetaya. Evaluating and calibrating uncertainty prediction in regression tasks. *Sensors*, 22(15), 2022b. ISSN 1424-8220. doi: 10.3390/s22155540. URL https://www.mdpi.com/1424-8220/22/15/5540.

Wei Li, Ruqing Fang, Junning Jiao, Georgios N Vassilakis, and Juner Zhu. Tutorials: Physics-informed machine learning methods of computing 1d phase-field models. *APL Machine Learning*, 2(3), 2024.

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. *arXiv preprint arXiv:2206.11990*, 2022.

Yi-Lun Liao, Brandon Wood, Abhishek Das, and Tess Smidt. Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations. *arXiv preprint arXiv:2306.12059*, 2023.

Tobias Liaudat, Jean-Luc Starck, Martin Kilbinger, and Pierre-Antoine Frugier. Rethinking data-driven point spread function modeling with a differentiable optical model. *Inverse Problems*, 39(3):035008, 2023.

life4. Deal: A python library for design by contract programming. https://github.com/life4/deal, 2024. Accessed: 2024-09-07.

Lin, Nie, Huanyuan, Shan, Guoliang, Li, Lei, Wang, Charling, Tao, Qifan, Cui, Yushan, Xie, Dezi, Liu, Zekang, and Zhang. Hybpsf: Hybrid psf reconstruction for the observed jwst nircam image, 2023.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

Christopher Lourenco, Jinhao Chen, Erick Moreno-Centeno, and Timothy A. Davis. Algorithm 1021: Spex left lu, exactly solving sparse linear systems via a sparse left-looking integer-preserving lu factorization. *ACM Trans. Math. Softw.*, jun 2022. ISSN 0098-3500. doi: 10.1145/3519024. URL https://doi.org/10.1145/3519024.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Carlo Lucibello and other contributors. Graphneuralnetworks.jl: a geometric deep learning library for the julia programming language, 2021. URL https://github.com/CarloLucibello/GraphNeuralNetworks.jl.

Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. URL https://api.semanticscholar.org/CorpusID:16489696.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Rachel Mandelbaum, Christopher M Hirata, Uroš Seljak, Jacek Guzik, Nikhil Padmanabhan, Cullen Blake, Michael R Blanton, Robert Lupton, and Jonathan Brinkmann. Systematic errors in weak lensing: application to sdss galaxy-galaxy weak lensing. *Monthly Notices of the Royal Astronomical Society*, 361(4): 1287–1322, 2005.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pp. 4363–4371. PMLR, 2019.

Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. In *International conference on machine learning*, pp. 6734–6744. PMLR, 2020.

Sean Martin, Andrew M Raim, Wen Huang, and Kofi P Adragni. Manifoldoptim: an r interface to the roptlib library for riemannian manifold optimization. *Journal of Statistical Software*, 93:1–32, 2020.

Jacqueline McCleary, I Dell'Antonio, and P Huwe. Mass substructure in abell 3128. *The Astrophysical Journal*, 805(1):40, 2015.

Jacqueline McCleary, Ian dell'Antonio, and Anja von der Linden. Dark matter distribution of four low-z clusters of galaxies. *The Astrophysical Journal*, 893(1):8, 2020.

Jacqueline E. McCleary, Spencer W. Everett, Mohamed M. Shaaban, Ajay S. Gill, Georgios N. Vassilakis, Eric M. Huff, Richard J. Massey, Steven J. Benton, Anthony M. Brown, Paul Clark, Bradley Holder, Aurelien A. Fraisse, Mathilde Jauzac, William C. Jones, David Lagattuta, Jason S.-Y. Leung, Lun Li, Thuy Vy T. Luu, Johanna M. Nagy, C. Barth Netterfield, Emaad Paracha, Susan F. Redmond, Jason D. Rhodes, Jürgen Schmoll, Ellen Sirks, and Sut Ieng Tam. Lensing in the blue. ii. estimating the sensitivity of stratospheric balloons to weak gravitational lensing. *The Astronomical Journal*, 166(3):134, aug 2023a. doi: 10.3847/1538-3881/ace7ca. URL https://dx.doi.org/10.3847/1538-3881/ace7ca.

Jacqueline E McCleary, Spencer W Everett, Mohamed M Shaaban, Ajay S Gill, Georgios N Vassilakis, Eric M Huff, Richard J Massey, Steven J Benton, Anthony M Brown, Paul Clark, et al. Lensing in the blue ii: Estimating the sensitivity of stratospheric balloons to weak gravitational lensing. *arXiv preprint arXiv:2307.03295*, 2023b.

Wes McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

Leo A Meyerovich and Ariel S Rabkin. Empirical analysis of programming language adoption. In *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications*, pp. 1–18, 2013.

Kevin Michalewicz, Martin Millon, Frédéric Dux, and Frédéric Courbin. Starred: a two-channel deconvolution method with starlet regularization. *Journal of Open Source Software*, 8(85):5340, 2023. doi: 10.21105/joss.05340. URL https://doi.org/10.21105/joss.05340.

Microsoft. Pyright: Static type checker for python, 2023. URL https://microsoft.github.io/pyright/. Accessed: 2024-09-07.

Patrick K. Mogensen and Asbjørn N. Riseth. Optim: A mathematical optimization package for julia. *Journal of Open Source Software*, 3(24):615, 2018. doi: 10.21105/joss.00615. URL https://doi.org/10.21105/joss.00615.

William S. Moses, Valentin Churavy, Ludger Paehler, Jan Hückelheim, Sri Hari Krishna Narayanan, Michel Schanen, and Johannes Doerfert. Reverse-mode automatic differentiation and optimization of gpu kernels via enzyme. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3476165. URL https://doi.org/10.1145/3458817.3476165.

Radu Muschevici, Alex Potanin, Ewan Tempero, and James Noble. Multiple dispatch in practice. *Acm sigplan notices*, 43(10):563–582, 2008.

Dylan Nelson, Volker Springel, Annalisa Pillepich, Vicente Rodriguez-Gomez, Paul Torrey, Shy Genel, Mark Vogelsberger, Ruediger Pakmor, Federico Marinacci, Rainer Weinberger, et al. The illustristng simulations: public data release. *Computational Astrophysics and Cosmology*, 6:1–29, 2019.

Rebecca Nevin, Aleksandra Ćiprijanović, and Brian D Nord. Deepuq: Assessing the aleatoric uncertainties from two deep learning methods. *arXiv preprint arXiv:2411.08587*, 2024.

James. W. Nightingale, Richard G. Hayes, and Matthew Griffiths. 'pyautofit': A classy probabilistic programming language for model composition and fitting. *Journal of Open Source Software*, 6(58):2550, 2021. doi: 10.21105/joss.02550. URL https://doi.org/10.21105/joss.02550.

ROYUD Nishino and Shohei Hido Crissman Loomis. Cupy: A numpy-compatible library for nvidia gpu calculations. *31st confernce on neural information processing systems*, 151(7), 2017.

David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pp. 55–60. IEEE, 1994.

Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPR workshops*, volume 2, 2019.

Northeastern University Research Computing. Hardware overview. https://rc-docs.northeastern.edu/en/latest/hardware/hardware_overview.html, 2024. Accessed: 2024-01-20.

SciML Organization. Sciml roadmap, 2024. URL https://sciml.ai/roadmap/. Accessed: 2024-09-03.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Epistemic neural networks. *Advances in Neural Information Processing Systems*, 36:2795–2823, 2023.

Aayush Pal. Lux: Explicit parameterization of deep neural networks in julia (v0.4.50), 2023. URL https://doi.org/10.5281/zenodo.7808904.

Glenn Palmer, Siqi Du, Alexander Politowicz, Joshua Paul Emory, Xiyu Yang, Anupraas Gautam, Grishma Gupta, Zhelong Li, Ryan Jacobs, and Dane Morgan. Calibration after bootstrap for accurate uncertainty quantification in regression models. *npj Computational Materials*, 8(1):115, 2022.

Sneh Pandya, Purvik Patel, Jonathan Blazek, et al. E (2) equivariant neural networks for robust galaxy morphology classification. *arXiv preprint arXiv:2311.01500*, 2023.

Sneh Pandya, Yuanyuan Yang, Nicholas Van Alfen, Jonathan Blazek, and Robin Walters. Learning galaxy intrinsic alignment correlations. 2024.

Sneh Pandya, Purvik Patel, Brian D. Nord, Mike Walmsley, and Aleksandra Ćiprijanović. Sidda: Sinkhorn dynamic domain adaptation for image classification with equivariant neural networks, 2025a. URL https://arxiv.org/abs/2501.14048.

Sneh Pandya, Yuanyuan Yang, Nicholas Van Alfen, Jonathan Blazek, and Robin Walters. Iaemu: Learning galaxy intrinsic alignment correlations, 2025b. URL https://arxiv.org/abs/2504.05235.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Pascal Pernot. Properties of the ence and other mad-based calibration metrics. *arXiv preprint arXiv:2305.11905*, 2023.

Marshall D. Perrin, Rémi Soummer, Erin M. Elliott, Matthew D. Lallo, and Anand Sivaramakrishnan. Simulating point spread functions for the James Webb Space Telescope with WebbPSF. In Mark C. Clampin, Giovanni G. Fazio, Howard A. MacEwen, and Jr. Oschmann, Jacobus M. (eds.), *Space Telescopes and Instrumentation 2012: Optical, Infrared, and Millimeter Wave*, volume 8442 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 84423D, September 2012. doi: 10.1117/12.925230.

Marshall D. Perrin, Anand Sivaramakrishnan, Charles-Philippe Lajoie, Erin Elliott, Laurent Pueyo, Swara Ravindranath, and Loïc Albert. Updated point spread function simulations for JWST with WebbPSF. In Jacobus M. Oschmann Jr., Mark Clampin, Giovanni G. Fazio, and Howard A. MacEwen (eds.), *Space Telescopes and Instrumentation 2014: Optical, Infrared, and Millimeter Wave*, volume 9143, pp. 91433X. International Society for Optics and Photonics, SPIE, 2014a. doi: 10.1117/12.2056689. URL https://doi.org/10.1117/12.2056689.

Marshall D Perrin, Anand Sivaramakrishnan, Charles-Philippe Lajoie, Erin Elliott, Laurent Pueyo, Swara Ravindranath, and Loïc Albert. Updated point spread function simulations for jwst with webbpsf. In *Space telescopes and instrumentation 2014: optical, infrared, and millimeter wave*, volume 9143, pp. 1174–1184. SPIE, 2014b.

Mircea Petrache and Shubhendu Trivedi. Approximation-generalization trade-offs under (approximate) group equivariance. *Advances in Neural Information Processing Systems*, 36:61936–61959, 2023.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*, 2019.

Ava Polzin. spike: A tool to drizzle hst, jwst, and roman psfs for improved analyses, 2025. URL https://arxiv.org/abs/2503.02288.

Christopher Rackauckas and Qing Nie. Adaptive methods for stochastic differential equations via natural embeddings and rejection sampling with memory. *Discrete and continuous dynamical systems. Series B*, 22(7):2731, 2017a.

Christopher Rackauckas and Qing Nie. Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of open research software*, 5(1):15–15, 2017b.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Steven C. Rennich, Darko Stosic, and Timothy A. Davis. Accelerating sparse cholesky factorization on gpus. *Parallel Computing*, 59:140–150, 2016. ISSN 0167-8191. doi: https://doi.org/10.1016/j.parco.2016.06.004. URL https://www.sciencedirect.com/science/article/pii/S016781911630059X. Theory and Practice of Irregular Applications.

J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016. URL https://arxiv.org/abs/1607.07892.

Jarrett Revels, Mike Innes, Lyndon White, et al. Diffrules.jl. https://github.com/JuliaDiff/DiffRules.jl, 2018. Accessed: 2024-10-13.

Marcia J. Rieke, Stefi A. Baum, Charles A. Beichman, David Crampton, Rene Doyon, Daniel Eisenstein, Thomas P. Greene, Klaus-Werner Hodapp, Scott D. Horner, Doug Johnstone, Lawrence Lesyna, Simon Lilly, Michael Meyer, Peter Martin, Jr. McCarthy, Donald W., George H. Rieke, Thomas L. Roellig, John Stauffer, John T. Trauger, and Erick T. Young. NGST NIRCam Scientific Program and Design Concept. In John C. Mather (ed.), *IR Space Telescopes and Instruments*, volume 4850 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 478–485, March 2003. doi: 10.1117/12.489103.

Marcia J. Rieke, Douglas Kelly, and Scott Horner. Overview of James Webb Space Telescope and NIRCam's Role. In James B. Heaney and Lawrence G. Burriesci (eds.), *Cryogenic Optical Systems and Instruments XI*, volume 5904 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 1–8, August 2005. doi: 10.1117/12.615554.

Christian P Robert, George Casella, Christian P Robert, and George Casella. The metropolis—hastings algorithm. *Monte Carlo statistical methods*, pp. 267–320, 2004.

Brant E Robertson. Galaxy formation and reionization: key unknowns and expected breakthroughs by the james webb space telescope. *Annual Review of Astronomy and Astrophysics*, 60:121–158, 2022.

Thomas P Robitaille, Erik J Tollerud, Perry Greenfield, Michael Droettboom, Erik Bray, Tom Aldcroft, Matt Davis, Adam Ginsburg, Adrian M Price-Whelan, Wolfgang E Kerzendorf, et al. Astropy: A community python package for astronomy. *Astronomy & Astrophysics*, 558:A33, 2013.

David M Rosen. Accelerating certifiable estimation with preconditioned eigensolvers. *IEEE Robotics and Automation Letters*, 7(4):12507–12514, 2022.

David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019a. doi: 10.1177/0278364918784361. URL https://doi.org/10.1177/0278364918784361.

David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019b.

David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. A certifiably correct algorithm for synchronization over the special euclidean group. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pp. 64–79. Springer, 2020.

Barnaby Rowe. Improving psf modelling for weak gravitational lensing using new methods in model selection. *Monthly Notices of the Royal Astronomical Society*, 404(1):350–366, 2010.

Barnaby Rowe, Mike Jarvis, Rachel Mandelbaum, Gary M. Bernstein, James Bosch, Melanie Simet, Joshua E. Meyers, Tomasz Kacprzak, Reiko Nakajima, Joe Zuntz, Hironao Miyatake, Joerg P. Dietrich, Robert Armstrong, Peter Melchior, and Mandeep S. S. Gill. Galsim: The modular galaxy image simulation toolkit, 2015.

Barbara Ryden. *Introduction to cosmology*. Cambridge University Press, 2016.

Yousef Saad. Iterative methods for sparse linear systems.

Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

Andrew Sabot, Vikas Natesh, H. T. Kung, and Wei-Te Ting. Mema runtime framework: Minimizing external memory accesses for tinyml on microcontrollers, 2023.

Akiyoshi Sannai, Masaaki Imaizumi, and Makoto Kawano. Improved generalization bounds of group invariant/equivariant deep networks via quotient feature spaces. In *Uncertainty in artificial intelligence*, pp. 771–780. PMLR, 2021.

M. T. Sargent, C. M. Carollo, S. J. Lilly, C. Scarlata, R. Feldmann, P. Kampczyk, A. M. Koekemoer, N. Scoville, J. P. Kneib, A. Leauthaud, R. Massey, J. Rhodes, L. A. M. Tasca, P. Capak, H. J. McCracken, C. Porciani, A. Renzini, Y. Taniguchi, D. J. Thompson, and K. Sheth. The Evolution of the Number Density of Large Disk Galaxies in COSMOS. **??***jnlApJS*, 172(1):434–455, September 2007. doi: 10.1086/516584.

Andrew K Saydjari and Douglas P Finkbeiner. Photometry on structured backgrounds: Local pixel-wise infilling by regression. *The Astrophysical Journal*, 933(2):155, 2022.

Gabriele Scalia, Colin A Grambow, Barbara Pernici, Yi-Pei Li, and William H Green. Evaluating scalable uncertainty estimation methods for deep learning-based molecular property prediction. *Journal of chemical information and modeling*, 60(6):2697–2717, 2020.

C. Scarlata, C. M. Carollo, S. Lilly, M. T. Sargent, R. Feldmann, P. Kampczyk, C. Porciani, A. Koekemoer, N. Scoville, J. P. Kneib, A. Leauthaud, R. Massey, J. Rhodes, L. Tasca, P. Capak, C. Maier, H. J. McCracken, B. Mobasher, A. Renzini, Y. Taniguchi, D. Thompson, K. Sheth, M. Ajiki, H. Aussel, T. Murayama, D. B. Sanders, S. Sasaki, Y. Shioya, and M. Takahashi. COSMOS Morphological Classification with the Zurich Estimator of Structural Types (ZEST) and the Evolution Since z = 1 of the Luminosity Function of Early, Disk, and Irregular Galaxies. **??***jnlApJS*, 172(1):406–433, September 2007. doi: 10.1086/516582.

Frank Schäfer, Mohamed Tarek, Lyndon White, and Chris Rackauckas. Abstractdifferentiation. jl: Backend-agnostic differentiable programming in julia. *NeurIPS 2021 Differentiable Programming Workshop*, 2021.

Philip Schanely. Crosshair: An analysis tool for python that uses symbolic execution. https://github.com/pschanely/CrossHair, 2024. Accessed: 2024-09-07.

Chad Scherrer and Moritz Schauer. Applied measure theory for probabilistic modeling. *Proceedings of the JuliaCon Conferences*, 1(1):92, 2022. doi: 10.21105/jcon.00092. URL https://doi.org/10.21105/jcon.00092.

Conner Scherrer and contributors. Soss.jl, 2024. URL https://github.com/cscherrer/Soss.jl. GitHub repository.

Peter Schneider, Ludovic van Waerbeke, Martin Kilbinger, and Yannick Mellier. Analysis of two-point statistics of cosmic shear-i. estimators and covariances. *Astronomy & Astrophysics*, 396(1):1–19, 2002.

Peter Schneider, Christopher Kochanek, and Joachim Wambsganss. *Gravitational lensing: strong, weak and micro: Saas-Fee advanced course 33*, volume 33. Springer Science & Business Media, 2006.

Diana Scognamiglio. Exploring Cosmic Matter with Weak Gravitational Lensing in COSMOS-Web. *Bulletin of the AAS*, 56(2), feb 7 2024. https://baas.aas.org/pub/2024n2i149p06.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. *arXiv preprint arXiv:2203.09168*, 2022.

Yu Shi, Shuxin Zheng, Guolin Ke, Yifei Shen, Jiacheng You, Jiyan He, Shengjie Luo, Chang Liu, Di He, and Tie-Yan Liu. Benchmarking graphormer on large-scale molecular modeling datasets. *arXiv preprint arXiv:2203.04810*, 2022.

Noah Shinn. Solvent. https://github.com/noahshinn/solvent, 2024. Accessed: 2024-09-04.

Ashwin Prasad Shivarpatna Venkatesh, Samkutty Sabu, Jiawei Wang, Amir M. Mir, Li Li, and Eric Bodden. Typeevalpy: A micro-benchmarking framework for python type inference tools. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pp. 49–53, 2024.

Rémi Soummer, André Ferrari, Claude Aime, and Laurent Jolissaint. Speckle noise and dynamic range in coronagraphic images. *The Astrophysical Journal*, 669(1):642, 2007.

Volker Springel. The cosmological simulation code gadget-2. *Monthly notices of the royal astronomical society*, 364(4):1105–1134, 2005.

Sophia Huiwen Sun, Robin Walters, Jinxi Li, and Rose Yu. Probabilistic symmetry for multi-agent dynamics. In *Learning for Dynamics and Control Conference*, pp. 1231–1244. PMLR, 2023.

Romain Teyssier. Cosmological hydrodynamics with adaptive mesh refinement-a new high resolution code called ramses. *Astronomy & Astrophysics*, 385(1):337–364, 2002.

The Julia Language. Sparsearrays module. https://docs.julialang.org/en/v1/stdlib/SparseArrays/, 2024. Accessed: 2024-09-01.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds, 2018. URL https://arxiv.org/abs/1802.08219.

Marcin Tomczak, Siddharth Swaroop, and Richard Turner. Efficient low rank gaussian variational inference for neural networks. *Advances in Neural Information Processing Systems*, 33:4610–4622, 2020.

James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016.

Yueh-Hua Tu. Geometricflux. jl: a geometric deep learning library in julia. *Proceedings of JuliaCon*, 1:1, 2020.

Dennis Ulmer, Christian Hardmeier, and Jes Frellsen. Prior and posterior networks: A survey on evidential deep learning methods for uncertainty estimation. *arXiv preprint arXiv:2110.03051*, 2021.

Matias Valdenegro-Toro and Daniel Saromo Mori. A deeper look into aleatoric and epistemic uncertainty disentanglement. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1508–1516. IEEE, 2022.

Nicholas Van Alfen, Duncan Campbell, Jonathan Blazek, C Danielle Leonard, Francois Lanusse, Andrew Hearin, and Rachel Mandelbaum. An empirical model for intrinsic alignments: Insights from cosmological simulations. *The Open Journal of Astrophysics*, 7, 2024.

Putri A Van der Linden, Alexander Timans, Dharmesh Tailor, and Erik J Bekkers. On equivariant model selection through the lens of uncertainty. *arXiv preprint arXiv:2506.18629*, 2025.

Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 23(176):88–93, 1975.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Mark Vogelsberger, Jesus Zavala, Francis-Yan Cyr-Racine, Christoph Pfrommer, Torsten Bringmann, and Kris Sigurdson. Ethos–an effective theory of structure formation: dark matter physics as a possible explanation of the small-scale cdm problems. *Monthly Notices of the Royal Astronomical Society*, 460(2): 1399–1416, 2016.

Mark Vogelsberger, Federico Marinacci, Paul Torrey, and Ewald Puchwein. Cosmological simulations of galaxy formation. *Nature Reviews Physics*, 2(1):42–66, 2020.

Mike Walmsley, Micah Bowles, Anna MM Scaife, Jason Shingirai Makechemu, Alexander J Gordon, Annette Ferguson, Robert G Mann, James Pearson, Jürgen J Popp, Jo Bovy, et al. Scaling laws for galaxy images. *arXiv preprint arXiv:2404.02973*, 2024.

Dian Wang, Mingxi Jia, Xupeng Zhu, Robin Walters, and Robert Platt. On-robot learning with equivariant models. *arXiv preprint arXiv:2203.04923*, 2022a.

Dian Wang, Robin Walters, Xupeng Zhu, and Robert Platt. Equivariant $q$ learning in spatial action spaces. In *Conference on Robot Learning*, pp. 1713–1723. PMLR, 2022b.

Dian Wang, Jung Yeon Park, Neel Sortur, Lawson L.S. Wong, Robin Walters, and Robert Platt. The surprising effectiveness of equivariant models in domains with latent symmetry. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=P4MUGRM4Acu.

Dian Wang, Xupeng Zhu, Jung Yeon Park, Mingxi Jia, Guanang Su, Robert Platt, and Robin Walters. A general theory of correct, incorrect, and extrinsic equivariance. *Advances in Neural Information Processing Systems*, 36, 2024.

Jason J Wang, Jean-Baptise Ruffio, Robert J De Rosa, Jonathan Aguilar, Schuyler G Wolff, and Laurent Pueyo. pyklip: Psf subtraction for exoplanets and disks. *Astrophysics Source Code Library*, pp. ascl–1506, 2015.

Yuyang Wang, Ahmed AA Elhag, Navdeep Jaitly, Joshua M Susskind, and Miguel Ángel Bautista. Swallowing the bitter pill: Simplified scalable conformer generation. In *Forty-first International Conference on Machine Learning*, 2023b.

J. R. Weaver, O. B. Kauffmann, O. Ilbert, H. J. McCracken, A. Moneti, S. Toft, G. Brammer, M. Shuntov, I. Davidzon, B. C. Hsieh, C. Laigle, A. Anastasiou, C. K. Jespersen, J. Vinther, P. Capak, C. M. Casey, C. J. R. McPartland, B. Milvang-Jensen, B. Mobasher, D. B. Sanders, L. Zalesky, S. Arnouts, H. Aussel, J. S. Dunlop, A. Faisst, M. Franx, L. J. Furtak, J. P. U. Fynbo, K. M. L. Gould, T. R. Greve, S. Gwyn, J. S. Kartaltepe, D. Kashino, A. M. Koekemoer, V. Kokorev, O. Le Fèvre, S. Lilly, D. Masters, G. Magdis, V. Mehta, Y. Peng, D. A. Riechers, M. Salvato, M. Sawicki, C. Scarlata, N. Scoville, R. Shirley, J. D. Silverman, A. Sneppen, V. Smolčić, C. Steinhardt, D. Stern, M. Tanaka, Y. Taniguchi, H. I. Teplitz, M. Vaccari, W. H. Wang, and G. Zamorani. COSMOS2020: A Panchromatic View of the Universe to z 10 from Two Complementary Catalogs. **??***jnlApJS*, 258(1):11, January 2022. doi: 10.3847/1538-4365/ac3078.

Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. *Advances in neural information processing systems*, 32, 2019.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data, 2018. URL https://arxiv.org/abs/1807.02547.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Lisa Wimmer, Yusuf Sale, Paul Hofman, Bernd Bischl, and Eyke Hüllermeier. Quantifying aleatoric and epistemic uncertainty in machine learning: Are conditional entropy and mutual information appropriate measures? In *Uncertainty in Artificial Intelligence*, pp. 2282–2292. PMLR, 2023.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Tianshi Xu, Vassilis Kalantzis, Ruipeng Li, Yuanzhe Xi, Geoffrey Dillon, and Yousef Saad. pargemslr: A parallel multilevel schur complement low-rank preconditioning and solution package for general sparse matrices. *Parallel Computing*, 113:102956, 2022.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1):407–474, 2022.

Sencer Nuri Yeralan, Timothy A. Davis, Wissam M. Sid-Lakhdar, and Sanjay Ranka. Algorithm 980: Sparse qr factorization on the gpu. *ACM Trans. Math. Softw.*, 44(2), aug 2017. ISSN 0098-3500. doi: 10.1145/3065870. URL https://doi.org/10.1145/3065870.

Donald G. York, J. Adelman, Jr. John E. Anderson, Scott F. Anderson, James Annis, Neta A. Bahcall, J. A. Bakken, Robert Barkhouser, Steven Bastian, Eileen Berman, William N. Boroski, Steve Bracker, Charlie Briegel, John W. Briggs, J. Brinkmann, Robert Brunner, Scott Burles, Larry Carey, Michael A. Carr, Francisco J. Castander, Bing Chen, Patrick L. Colestock, A. J. Connolly, J. H. Crocker, István Csabai, Paul C. Czarapata, John Eric Davis, Mamoru Doi, Tom Dombeck, Daniel Eisenstein, Nancy Ellman, Brian R. Elms, Michael L. Evans, Xiaohui Fan, Glenn R. Federwitz, Larry Fiscelli, Scott Friedman, Joshua A. Frieman, Masataka Fukugita, Bruce Gillespie, James E. Gunn, Vijay K. Gurbani, Ernst de Haas, Merle Haldeman, Frederick H. Harris, J. Hayes, Timothy M. Heckman, G. S. Hennessy, Robert B. Hindsley, Scott Holm, Donald J. Holmgren, Chi hao Huang, Charles Hull, Don Husby, Shin-Ichi Ichikawa, Takashi Ichikawa, Željko Ivezić, Stephen Kent, Rita S. J. Kim, E. Kinney, Mark Klaene, A. N. Kleinman, S. Kleinman, G. R. Knapp, John Korienek, Richard G. Kron, Peter Z. Kunszt, D. Q. Lamb, B. Lee, R. French Leger, Siriluk Limmongkol, Carl Lindenmeyer, Daniel C. Long, Craig Loomis, Jon Loveday, Rich Lucinio, Robert H. Lupton, Bryan MacKinnon, Edward J. Mannery, P. M. Mantsch, Bruce Margon, Peregrine McGehee, Timothy A. McKay, Avery Meiksin, Aronne Merelli, David G. Monet, Jeffrey A. Munn, Vijay K. Narayanan, Thomas Nash, Eric Neilsen, Rich Neswold, Heidi Jo Newberg, R. C. Nichol, Tom Nicinski, Mario Nonino, Norio Okada, Sadanori Okamura, Jeremiah P. Ostriker, Russell Owen, A. George Pauls, John Peoples, R. L. Peterson, Donald Petravick, Jeffrey R. Pier, Adrian Pope, Ruth Pordes, Angela Prosapio, Ron Rechenmacher, Thomas R. Quinn, Gordon T. Richards, Michael W. Richmond, Claudio H. Rivetta, Constance M. Rockosi, Kurt Ruthmansdorfer, Dale Sandford, David J. Schlegel, Donald P. Schneider, Maki Sekiguchi, Gary Sergey, Kazuhiro Shimasaku, Walter A. Siegmund, Stephen Smee, J. Allyn Smith, S. Snedden, R. Stone, Chris Stoughton, Michael A. Strauss, Christopher Stubbs, Mark SubbaRao, Alexander S. Szalay, Istvan Szapudi, Gyula P. Szokoly, Anirudda R. Thakar, Christy Tremonti, Douglas L. Tucker, Alan Uomoto, Dan Vanden Berk, Michael S. Vogeley, Patrick Waddell, Shu i Wang, Masaru Watanabe, David H. Weinberg, Brian Yanny, and Naoki Yasuda. The sloan digital sky survey: Technical summary. *The Astronomical Journal*, 120(3):1579, sep 2000. doi: 10.1086/301513. URL https://dx.doi.org/10.1086/301513.

Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

Richard Zhang. Making convolutional networks shift-invariant again. In *International conference on machine learning*, pp. 7324–7334. PMLR, 2019.

Tianqing Zhang, Husni Almoubayyed, Rachel Mandelbaum, Joshua E Meyers, Mike Jarvis, Arun Kannawadi, Morgan A Schmitz, Axel Guinot, and LSST Dark Energy Science Collaboration. Impact of point spread function higher moments error on weak gravitational lensing–ii. a comprehensive study. *Monthly Notices of the Royal Astronomical Society*, 520(2):2328–2350, 2023.

Zheng Zheng, Alison L. Coil, and Idit Zehavi. Galaxy evolution from halo occupation distribution modeling of deep2 and sdss galaxy clustering. *The Astrophysical Journal*, 667(2):760, oct 2007. doi: 10.1086/521074. URL https://dx.doi.org/10.1086/521074.

Zihan Zou, Yujin Zhang, Lijun Liang, Mingzhi Wei, Jiancai Leng, Jun Jiang, Yi Luo, and Wei Hu. A deep learning model for predicting selected organic molecular spectra. *Nature Computational Science*, 3:1–8, 11 2023a. doi: 10.1038/s43588-023-00550-y.

Zihan Zou, Yujin Zhang, Lijun Liang, Mingzhi Wei, Jiancai Leng, Jun Jiang, Yi Luo, and Wei Hu. A deep learning model for predicting selected organic molecular spectra. *Nature Computational Science*, 3(11): 957–964, 2023b.

Kirill Zubov, Zoe McCarthy, Yingbo Ma, Francesco Calisto, Valerio Pagliarino, Simone Azeglio, Luca Bottero, Emmanuel Luján, Valentin Sulzer, Ashutosh Bharambe, Nand Vinchhi, Kaushik Balakrishnan, Devesh Upadhyay, and Chris Rackauckas. Neuralpde: Automating physics-informed neural networks (pinns) with error approximations, 2021. URL https://arxiv.org/abs/2107.09443.

# Part VIII

# Part I Appendix

## A PSF Modeling

This analysis uses ShOpt.jl for PSF modeling because it was shown to be accurate and computationally fast in Berman et al. (2024); Berman & McCleary (2024a). Since we are expanding our input catalog to the entire survey area of the COSMOS field instead of individual tiles as in Berman et al. (2024), we also expect astrometric distortions to be more severe (cf. Figure 4 in Berman et al. (2024)). Since ShOpt works in astrometric coordinates and is compatible with PSF cutout sizes needed for NIRCam, ShOpt is well-suited for this task. In contrast, PSFex exhibits bias over large survey areas (Jarvis et al., 2021) and PIFF does not handle cutout sizes needed for characterizing the NIRCam PSF gracefully (Berman et al., 2024). Tools like WebbPSF (Perrin et al., 2014b) were not designed for the mosaiced images we are working with, making their application to our data non trivial. Other tools like STARRED (Michalewicz et al., 2023) or PSFr (Birrer et al. in prep, Birrer et al., 2021) have been applied to JWST data and could potentially also be applied to this data set. See also Feng et al. (2025).

## B $\rho$ statistics

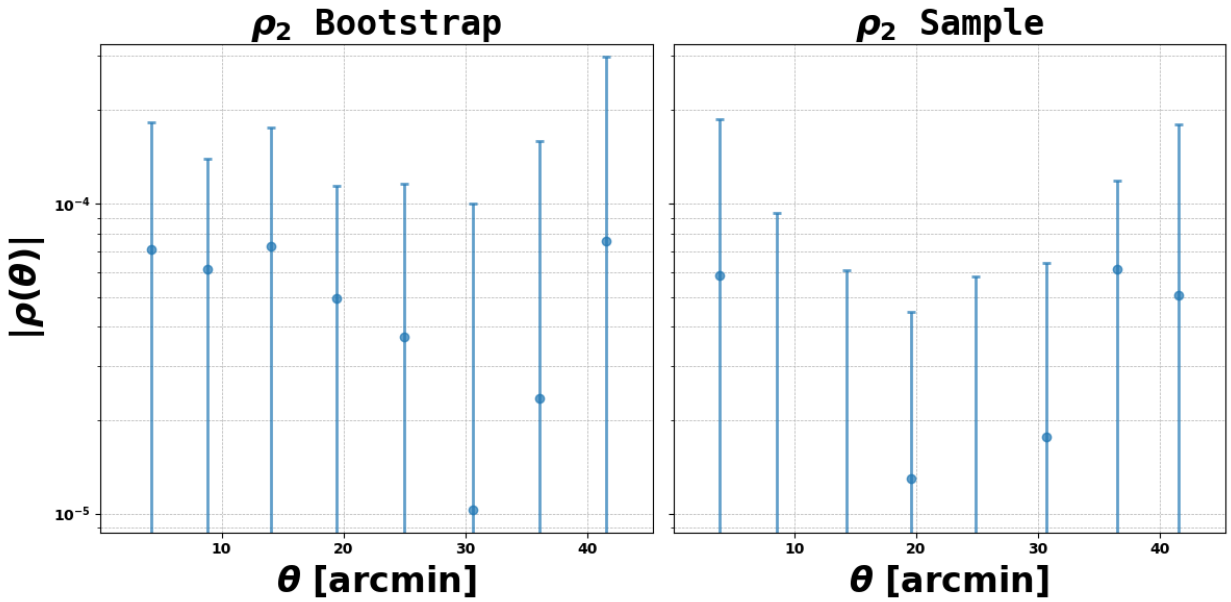



Figure 44: $\rho_2$ correlation function with data taken from the COSMOS-Web point source catalogs. Error bars drawn from bootstrap (left) and sampling (right) approaches. Negative correlations are shown in absolute value, and correlations are plotted in logarithmic scale.

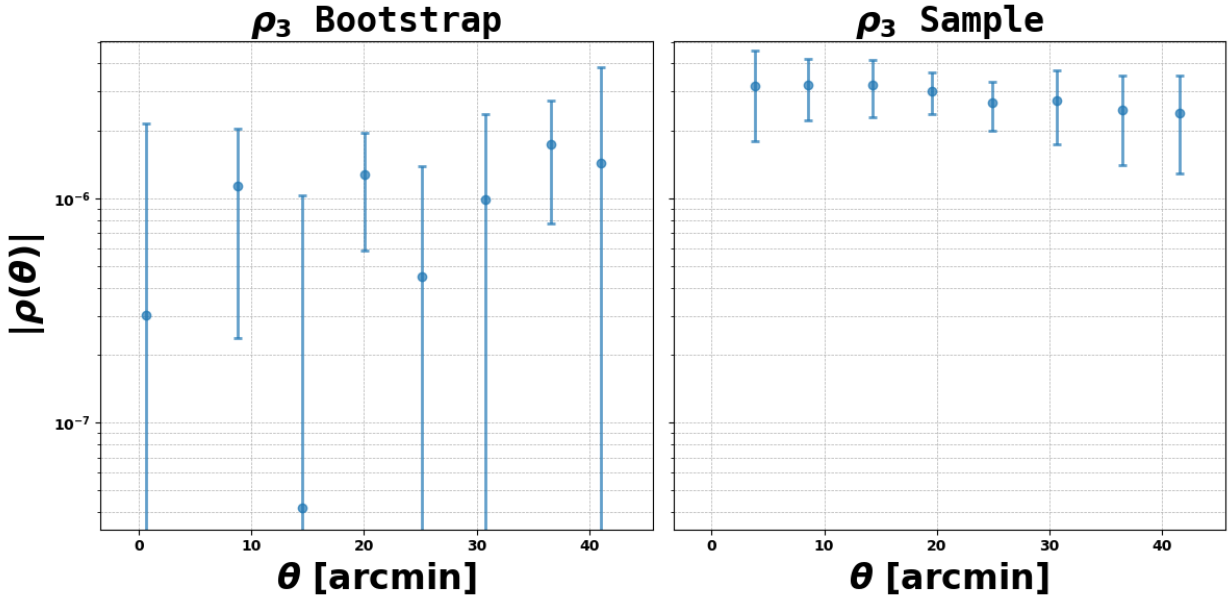



Figure 45: $\rho_3$ correlation function with data taken from the COSMOS-Web point source catalogs. Error bars drawn from bootstrap (left) and sampling (right) approaches. Negative correlations are shown in absolute value, and correlations are plotted in logarithmic scale.

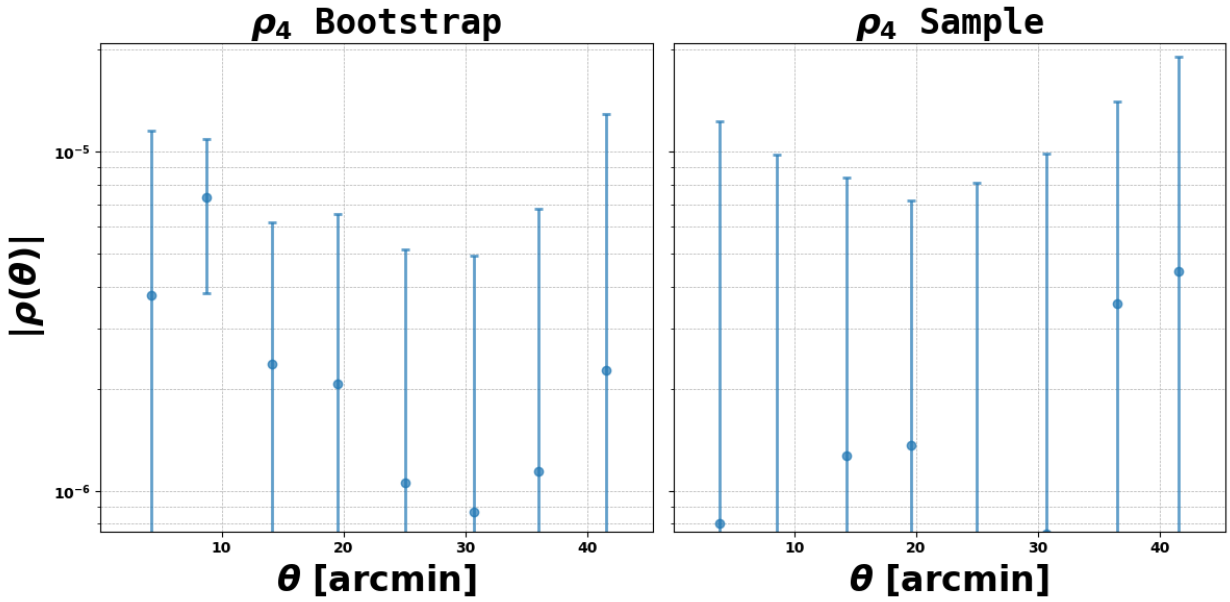



Figure 46: $\rho_4$ correlation function with data taken from the COSMOS-Web point source catalogs. Error bars drawn from bootstrap (left) and sampling (right) approaches. Negative correlations are shown in absolute value, and correlations are plotted in logarithmic scale.

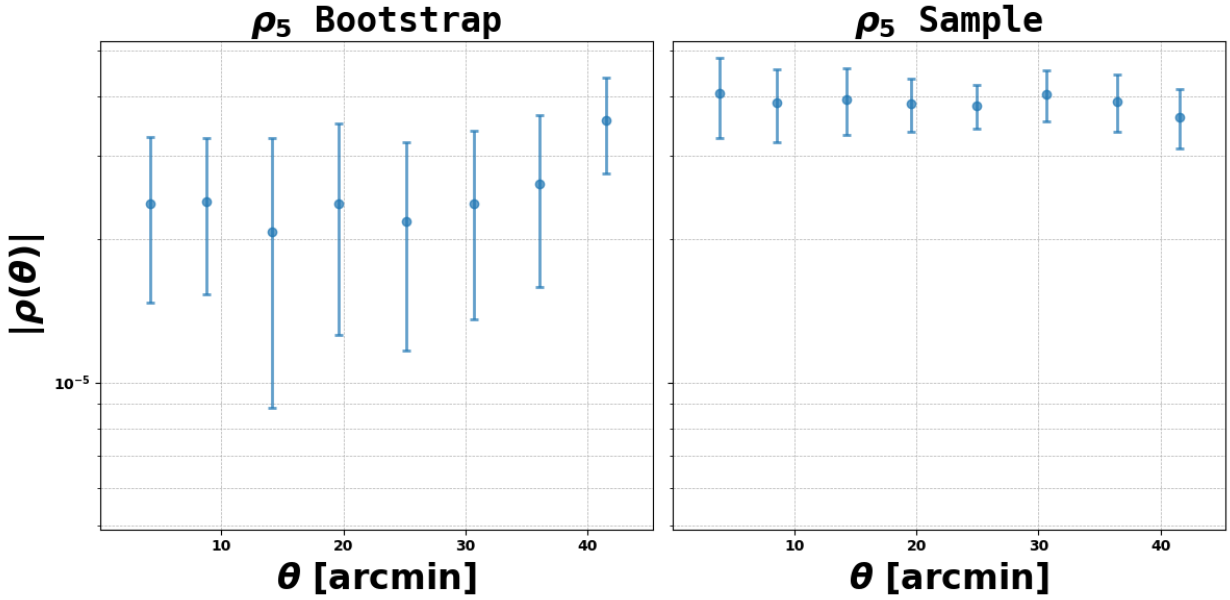



Figure 47: $\rho_5$ correlation function with data taken from the COSMOS-Web point source catalogs. Error bars drawn from bootstrap (left) and sampling (right) approaches. Negative correlations are shown in absolute value, and correlations are plotted in logarithmic scale.

## C Jacobians

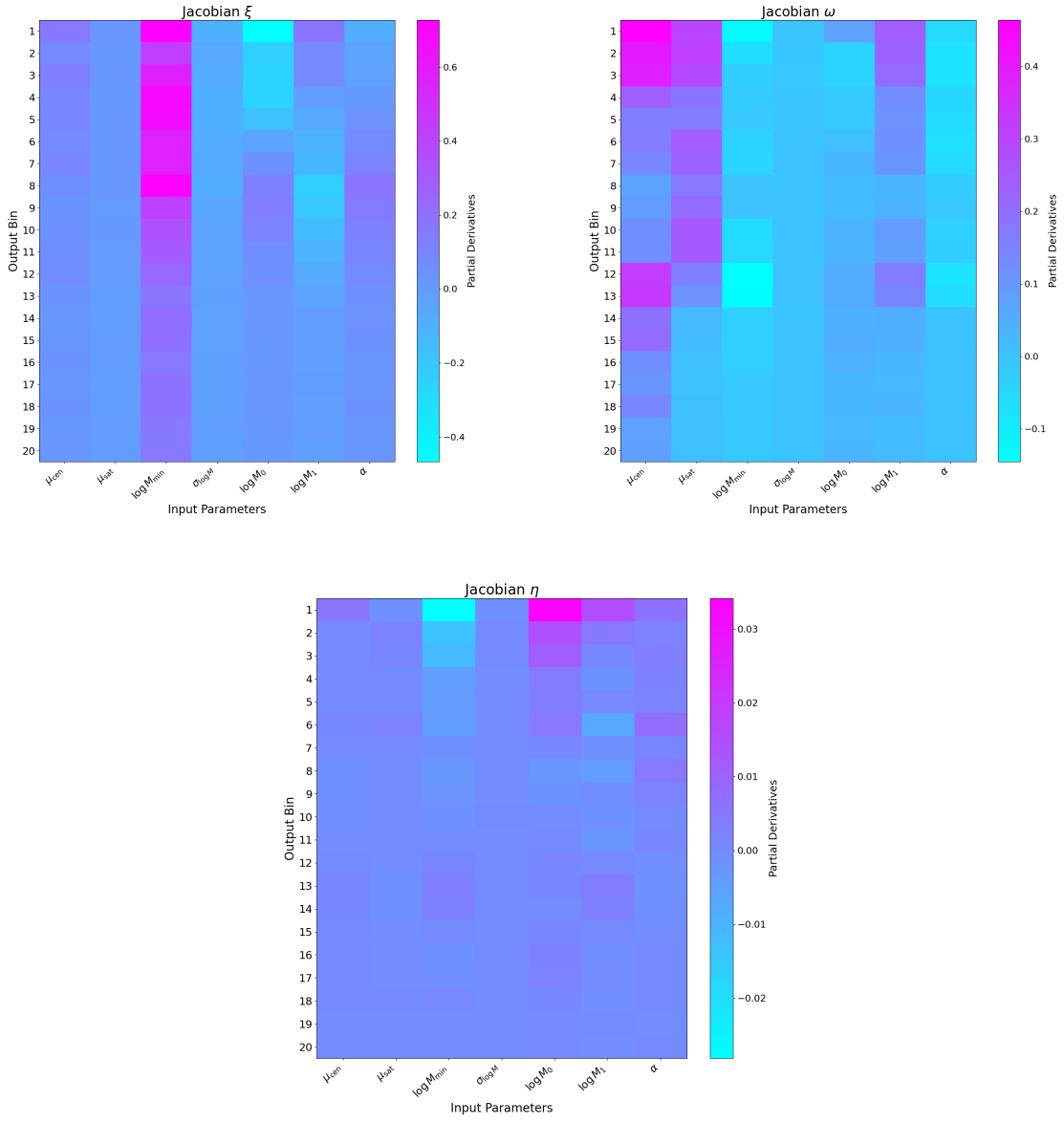



Figure 48: Jacobians associated with each of the 7 IA parameters and 20 output bins from IA-Emu.

# Part IX

# Part III Appendix

## A Iterated Integration

**Definition 9** (**Definition 4.1 in Wang et al. (2024)**). Let $d$ be the dimension of a generic orbit of $G$ in $X$ and $n$ the dimension of $X$. Let $\nu$ be the $(n-d)$ dimensional Hausdorff measure in $X$. A closed subset $F$ of $X$ is called a fundamental domain of $G$ in $X$ if $X$ is the union of conjugates of $F$, i.e., $X = \cup_{g\in G} gF$, and the intersection of any two conjugates has 0 measure under $\nu$.

If we assume that $\cup_{g_1\neq g_2}(g_1F \cap g_2F)$ has measure 0 and $F$ and $Gx$ are differentiable manifolds, then we may lift an integral $Gx$ to itself. Denote the identification of the orbit $Gx$ and coset space $G/G_x$ with respect to the stabilizer $G_x = \{g : gx = x\}$ by $a_x : G/G_x \to Gx$. Then we have

$$\int_{Gx} f(z)dz = \int_G f(gx)\alpha(g,x)dg \tag{110}$$

where

$$\alpha(g,x) = \left(\int_{Gx} dh\right)^{-1} \left| \frac{\partial a_x(\bar{g})}{\partial \bar{g}} \right|. \tag{111}$$

## B Mean-Invariant Coverage Problem Setup

In many practical circumstances, we will be working with a model $h$ that has different equivariance constraints for the prediction $m$ and variance $s$. Here, we consider a class of models $\{h : X \to \boldsymbol{\mathcal{M}} \times \boldsymbol{\mathcal{S}}\}$ where $h_\mu$ is $G-$invariant but $h_{\sigma^2}$ is a constant function. Let $\hat{C}^{1-\alpha}$ be a $1-\alpha$ confidence interval for $1-\alpha \in [0,1]$. Our goal for our model is to maximize the expected coverage, where coverage is defined as in Sun et al. (2023):

$$\text{Coverage} = \mathbb{I}\left(\mathbb{P}(f \in \hat{C}^{1-\alpha}) \geq 1-\alpha\right). \tag{112}$$

Accordingly the expected coverage is

$$\mathbb{E}_{x\sim p}\left[\mathbb{I}\left(\mathbb{P}(f(x) \in \hat{C}^{1-\alpha}) \geq 1-\alpha\right)\right] = \int_F \int_{Gx} p(z)\left(\mathbb{I}\left(\mathbb{P}(f(x) \in \hat{C}^{1-\alpha}) \geq 1-\alpha\right)\right) dzdx. \tag{113}$$

The expression $\mathbb{P}(f \in \hat{C}^{1-\alpha})$ can be formulated in terms of the Mahalanobis distance:

$$d^2(h_\mu(x), f(x)) = -\left[h_\mu(x) - f(x)\right]^T \Sigma^{-1} \left[h_\mu(x) - f(x)\right] \tag{114}$$

where $\Sigma^{-1} = \text{diag}(h_{\sigma^2})$. For brevity, we will sometimes just abbreviate $d^2$ or $d^2(x)$ instead of $d^2(h_\mu(x), f(x))$. Additionally, we will note that the distance is symmetric in that

$$d^2(h_\mu(x), f(x)) = d^2(f(x), h_\mu(x)). \tag{115}$$

It is known that $d^2$ follows a $\chi^2$ distribution, which has CDF[19] given by

$$\gamma(s,w) = \int_0^w t^{s-1}e^{-t}dt \tag{116}$$

where $s$ is the degrees of freedom. The degrees of freedom $s$ is taken to be $\dim(Y)$. If $\gamma(s,w^*) = 1-\alpha$, then the $1-\alpha$ confidence interval is given by $\gamma^{-1}(1-\alpha)$. We know that $\gamma$ is invertible because it is monotonically increasing in $w$.

Now, we can rewrite coverage via

[19] For more on the $\chi^2$ CDF, see Beh (2018)

$$\mathbb{I}\left(\mathbb{P}(f \in \hat{C}^{1-\alpha}) \geq 1-\alpha\right) = \mathbb{I}(d^2 \leq \gamma^{-1}(1-\alpha)). \tag{117}$$

The expected coverage then becomes

$$\mathbb{E}_{x\sim p}[\text{Coverage}] = \int_F \int_{Gx} p(z)\left[\mathbb{I}(d^2 \leq \gamma^{-1}(1-\alpha))\right] dzdx. \tag{118}$$

One may try to take advantage of invariance to bound the Coverage metric by placing a bound on the distance. However, on a given orbit $Gx$, the best fit function $h_\mu$ is the average of the function $f$ on the orbit defined. Therefore, $h_\mu$ is usually equal to $f$ on at least one input $x$, making the lower bound on $d^2$ become 0, and the assumption of invariance less than useful. Future work may explore this in a manner closer to Lemos et al. (2023) with the treatment of credible regions and expected coverage.

## C Special Case for Invariant Approximation Error

**Corollary 3.** A function $f$ is said to be uniformly continuous if for some $\delta > 0$ and for any $x$, $y \in \mathbb{R}$ we have that $|x-y| < \delta \implies |f(x) - f(y)| < \varepsilon$ for all $\varepsilon > 0$. Let $G$ be a group and let $h$ be a $G-$invariant function. Let $\Pi : [a,b] \to [a,b]$ be a function that rearranges the elements $x \in [a,b]$ so that the orbits $Gx$ are arranged sequentially, let $\widetilde{f}$ be given by $f(\Pi^{-1}(x))$, and assume the following conditions:

1. The domain of $\widetilde{f}$ is $\Pi([a,b])$ for some $a$, $b \in \mathbb{R}$. The co-domain is $\mathbb{R}$.
2. $\widetilde{f}$ is continuous on $\Pi([a,b])$.
3. For each orbit $Gx$, $\sup Gx \in Gx$, $\inf Gx \in Gx$.
4. For each orbit $Gx$, $|\sup Gx - \inf Gx| < \delta$.

It follows that we can choose the $G-$invariant function $h$ such that $|\widetilde{f}(x) - h(x)| < \varepsilon$ for all $x \in \Pi([a,b])$.

*Proof.* It is known that continuous functions on closed a closed interval are uniformaly continuous. Now, we partition the domain $[a,b]$ into each of the orbits $Gx$ after applying $\Pi$. By intermediate value theorem, $\widetilde{f}$ attains its average on each orbit. On each orbit, we choose $y$ to be the real number that corresponds to $\widetilde{f}(y) = \frac{1}{\sup Gx - \inf Gx}\int_{z\in Gx} \widetilde{f}(z)dz$ on the orbit. Therefore, we have that for $x$, $y \in Gx$, $|x-y| < \delta \implies |\widetilde{f}(x) - \widetilde{f}(y)| < \varepsilon$, where $|x-y| < \delta$ is true by the fourth assumption. We set $h(x) \equiv \widetilde{f}(y)$ which gives us that $|\widetilde{f}(x) - h(x)| < \varepsilon$ for all $x \in \Pi([a,b])$. This completes the proof.

□

**Remark 7.** This proof is a special case of the more general fact that if $f$ is continuous on a closed interval $[a,b]$, then we can choose a piecewise constant function $h_\varepsilon(x)$ such that $|f(x) - h_\varepsilon(x)| < \varepsilon$. The significance of this corollary is that it tells us that if our orbits are sufficiently small in $\mathbb{R}$ then the $G-$invariant regression error can be made arbitrarily small too. If we partition $[a,b]$ into the orbits induced by $\sigma^2$, then we may use this to understand if the domain restricted regression error can be made arbitrarily small. We caution that the assumptions for this to work are quite strong. In particular, we assume that $f$ is a function $[a,b] \to \mathbb{R}$. The language of representation theory still applies, since fields are trivially vectors spaces over themselves (e.g. real numbers are vector spaces over themselves), however, we are still heavily constrained by this choice.

## D Vector Regression Setup

Our training of the $E(3)-$equivariant neural network uses e3nn_jax (Geiger et al., 2022; Geiger & Smidt, 2022; Kondor et al., 2018; Weiler et al., 2018; Thomas et al., 2018). The MLP baseline is built entirely with Flax (Heek et al., 2024). The models are trained for a minimum of 10 epochs for a maximum number of 100, with early stopping if the validation loss stops improving after 5 epochs. We train on 2000 generated samples. We train using both a $\beta-$NLL loss (Seitzer et al., 2022) and an MSE loss, equally weighted, with $\beta = 1$. The $\beta-$NLL loss is given by

$$\mathcal{L}_{\beta-NLL} := \mathbb{E}_{X,Y}\left[\lfloor\hat{\sigma}^{2\beta}\rfloor\left(\frac{1}{2}\log\hat{\sigma}^2 + \frac{(Y-\hat{\mu}(X))^2}{2\hat{\sigma}^2}\right) + C\right] \tag{119}$$

where $\lfloor\cdot\rfloor$ represents a stop-gradient. For consistency with the figures, the reported metrics are calculated on the $xy-$coordinates. The MSE and $\beta$-NLL scores are average over all vectors and $xy-$coordinates.

## E Swiss Roll Experiment Details

The Swiss Roll distributions are created by generating points in polar coordinates using some $r$ as a function of $\theta$. Additionally, the points are given a $z-$coordinate of 0 or 1. An example of a spiral distribution with extrinsic equivariance seen from a $z-$invariant point-of-view is given in Figure 49. See also Figures 7 and 11 in Wang et al. (2024). The correct and incorrect Swiss Roll Distributions are similar. For correct equivariance, the color labels are the same for each spiral at $z = 0$ and $z = 1$. For incorrect, the labels are the opposite. For extrinsic, the spirals do not overlap. For details further, see Wang et al. (2024).

Can i just copy the figures from the previous paper and give a cite rather than directing someone to look at the figures in another paper? Can I copy the figures 7 and 11 i mention directly?

**Binning Approximations.** We compute ECE using the following binning approximations:

$$\text{acc}(B_m) = \frac{1}{|B_m|}\sum_{i\in B_m}\mathbb{I}(f = h_Y) \tag{120}$$

$$\text{conf}(B_m) = \frac{1}{|B_m|}\sum_{i\in B_m} h_P \tag{121}$$

$$\text{ECE} = \sum_{m=1}^{M}\frac{|B_m|}{n}\left|\,\text{acc}(B_m) - \text{conf}(B_m)\,\right|. \tag{122}$$

We use 100 bins. We adapt models, data generation, and training materials from Wang et al. (2024) and `https://github.com/pointW/ext_theory/`. The $z-$invariant network is implemented using DSS layers (Maron et al., 2020).

**Sample Calibration Approximation Error.** Theorem 3 tells us that ECE is bounded on a closed interval (regardless of any assumption of invariance). This allows us to say something about how many samples we need to approximate the true ECE. In particular, since ECE is bounded on $[0, 1]$, we may apply Hoeffding's Inequality (Hoeffding, 1994; 1963). We adopt notation similar to Bousquet et al. (2004); Petrache & Trivedi (2023). We sample $x$ according to $q(x)$ and obtain $h_Y$ and $h_p$. Define the calibration CE as the term inside the integrand of Equation 39, $|\mathbb{P}(f = h_Y|h_p = p) - p|$. For brevity, we will use $h(x) := (h_Y(x), h_P(x))$. The CE term still uses $h_Y(x)$ and $h_P(x)$ terms as distinct inputs. Imagine we sample $n$ times, giving us the set $\{(x_1, h(x_1)), \ldots, (x_n, h(x_n))\}$. We will abbreviate each i.i.d. $(x_i, h(x_i))$ pair as $Z_i$.

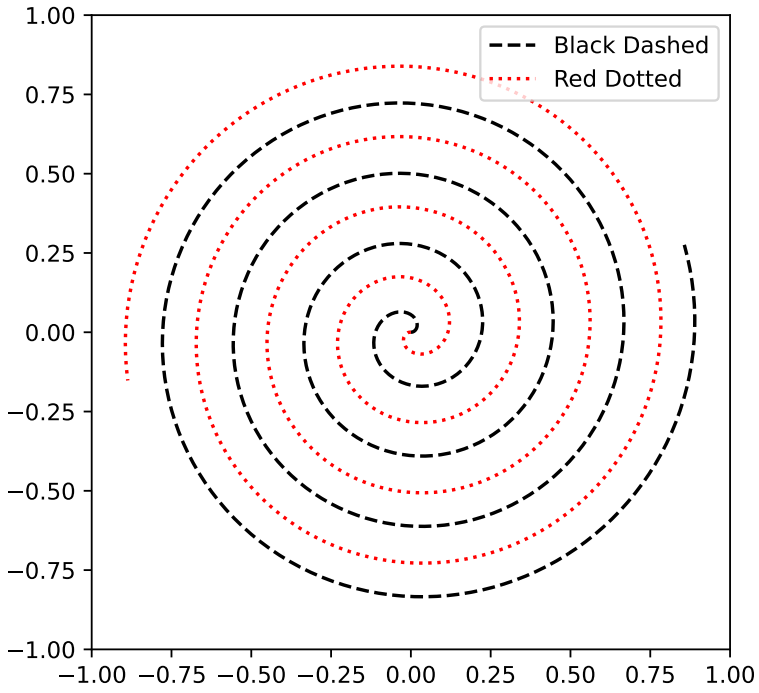



Figure 49: The extrinsic Swiss Roll Distribution seen from a $z-$invariant point of view.

**Corollary 4.**

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^{n} CE(Z_i) - ECE(Z)\right| > \varepsilon\right] \leq 2\exp(-2n\varepsilon^2) \tag{123}$$

for all $\varepsilon > 0$.

*Proof.* Since Theorem 3 tells us that ECE is bounded on $[0, 1]$, the result follows immediately from Hoeffding's Inequality. □

# F Galaxy Experiment Details

## F.1 Motivation and Implementation of PSF Blurring

**Motivation.** A point-spread function (PSF) is an impulse response of an optical system to light. PSFs occur all throughout medical and astronomical imaging. The science case we explore in this work is the distortion of galaxy images. With next generation imagers like JWST and large astronomical surveys like COSMOS-Web (Casey et al., 2023b), there are renewed efforts to characterize the effect of the PSF and its effects on downstream scientific analysis (Perrin et al., 2014b; Birrer et al., 2021; Jarvis et al., 2021; Michalewicz et al., 2023; Liaudat et al., 2023; Berman et al., 2024; Berman & McCleary, 2024a; Feng et al., 2025; Polzin, 2025). Understanding how the PSF harms a model's ability to identify a galaxy's morphology class can hint at the effect of the PSF on measured ellipticity moments (Hirata & Seljak, 2003; Mandelbaum et al., 2005), which is a crucial ingredient for maps of large scale structure (Scognamiglio, 2024). See, for example, McCleary et al. (2015; 2020).

**Implementation.** The way we implement PSF blurring follows Pandya et al. (2025a). Consider an image grid $I$ with values $(\zeta, \xi)$ and channels $c$. PSF blurring with a Gaussian kernel of width $\epsilon$ via

$$I_{\text{PSF}}(\zeta, \xi) = (I * G)(\zeta, \xi), \tag{124}$$

where

$$G(\zeta, \xi) = \frac{1}{2\pi\epsilon^2} \exp\left(-\frac{\zeta^2 + \xi^2}{2\epsilon^2}\right). \tag{125}$$

We apply this convolution on each channel $c$.

### F.2 Training and Evaluating

Our models and training scripts are adapted from Pandya et al. (2025a) and `https://github.com/deepskies/SIDDA`. The galaxy datasets are initially sourced from `https://zenodo.org/records/14583107`, and there is a script to produce the datasets with PSF blurring in our GitHub artifact. We compute ECE using the same approximations as in Equations 120 - 122. We summarize the number of parameters for each model in Table 8 below:

| Model | $CNN$ | $C_2$ | $C_4$ | $C_6$ | $C_8$ | $C_{10}$ | $C_{12}$ |
|---|---|---|---|---|---|---|---|
| Parameters | $1,188,486$ | $1,190,070$ | $1,197,750$ | $1,205,430$ | $1,213,110$ | $1,220,790$ | $1,228,470$ |
| Model | – | $D_2$ | $D_4$ | $D_6$ | $D_8$ | $D_{10}$ | $D_{12}$ |
| Parameters | – | $1,197,750$ | $1,213,110$ | $1,228,470$ | $1,243,830$ | $1,259,190$ | $1,274,550$ |

Table 8: Number of model parameters for Galaxy CNN and GCNN group order experiment.

For further guidance on how many hidden units are needed to approximate the ground truth as a function of group order, we direct the reader to Theorem 16 in Lawrence (2022).

## G Chemical Property Experiment Details

Our experiment for the chemical properties used a modified version of Backenköhler et al. (2023) for the data preprocessing and main training loop. While their analysis uses on feed forward network head for the prediction task, we use four independent feed forward heads that predict the quantities $m = (\gamma, \nu, \alpha, \beta)$. We train with a negative log likelihood loss function with an added regression loss regularizer,

$$\begin{aligned}
\Omega &= 2\beta(1+\nu) && (126)\\
\mathcal{L}_i^{\text{NLL}}(w) &= \frac{1}{2}\log\left(\frac{\pi}{\nu}\right) - \alpha\log(\Omega) && (127)\\
&+ \left(\alpha + \frac{1}{2}\right)\log((y_i - \gamma)^2\nu + \Omega) + \log\left(\frac{\Gamma(\alpha)}{\Gamma(\alpha + \frac{1}{2})}\right) && (128)\\
\mathcal{L}_i^{\text{R}}(w) &= |y_i - \mathbb{E}[\mu_i]| \cdot \Phi && (129)\\
&= |y_i - \gamma| \cdot (2\nu + \alpha) && (130)\\
\mathcal{L}_i(w) &= \mathcal{L}_i^{\text{NLL}}(w) + \lambda\mathcal{L}_i^{\text{R}}(w). && (131)
\end{aligned}$$

The GIN model has $52,417$ paramaters and the $E(3)-$invariant model has $51,969$. Through ablation study, we found that training stability is sensitive to a choice of $\lambda$, which we choose to be either $\lambda = 0.1$ or $\lambda = 1$. This instability is consistent with §S2.1.3 in Amini et al. (2020). Additionally, we found $z-$scoring the training, validation, and testing sets was necessary for ensuring stability during training for all molecular properties outside of the dipole moment.

Our model for emulating spectral lines is trained in the same way, partially taking inspiration from Zou et al. (2023b). We note the following tradeoffs between our approach and DetaNet:

1. Our adoption of the message-passing framework is more general than the attentional one used in their work (Bronstein et al., 2021).
2. DetaNet has arbitrary resolution, relying on a sum of basis functions.
3. DetaNet is trained not to produce the spectral line directly, but to produce the dipole moment, polarizability, and the inter-atomic and atomic hessians, which in turn gives the spectral line.
4. DetaNet can be limited by its usage of the Quantum Harmonic Oscillator approximation in some cases.

We leave further comparison and model development as an opportunity for future work. Other potential baselines could include Equiformer (Liao & Smidt, 2022), EquiformerV2 (Liao et al., 2023), Graphormer (Shi et al., 2022), or Graphormer with data augmentation. The model we use in this work has $36,997,125$ parameters.

## H Galaxy Experiment Additional Results

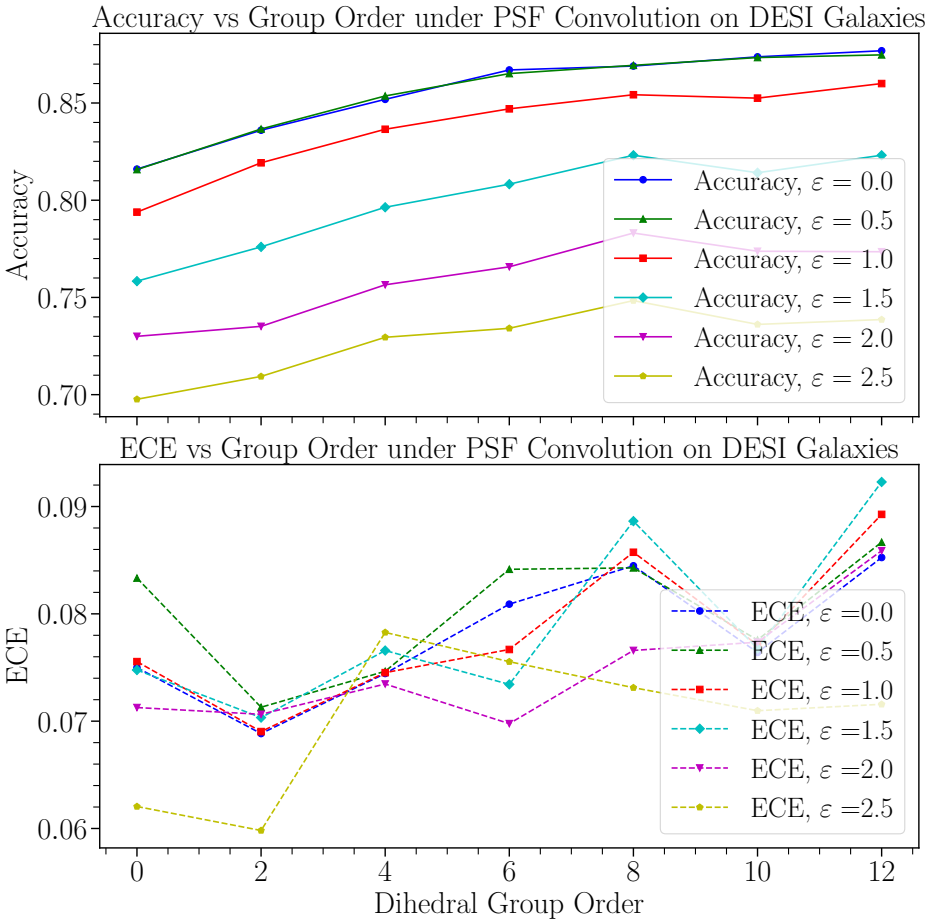



Figure 50: Accuracy and ECE vs Dihedral Group Order under PSF Convolution on DESI Galaxies.

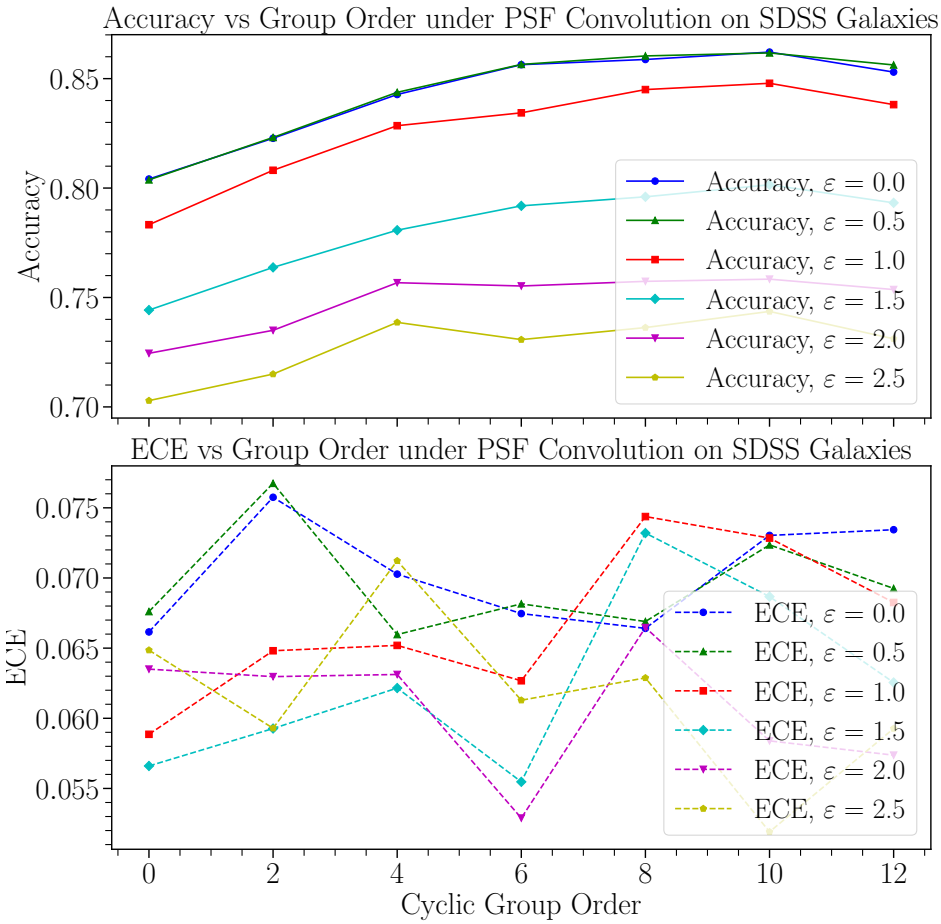



Figure 51: Accuracy and ECE vs Cyclic Group Order under PSF Convolution on SDSS Galaxies.

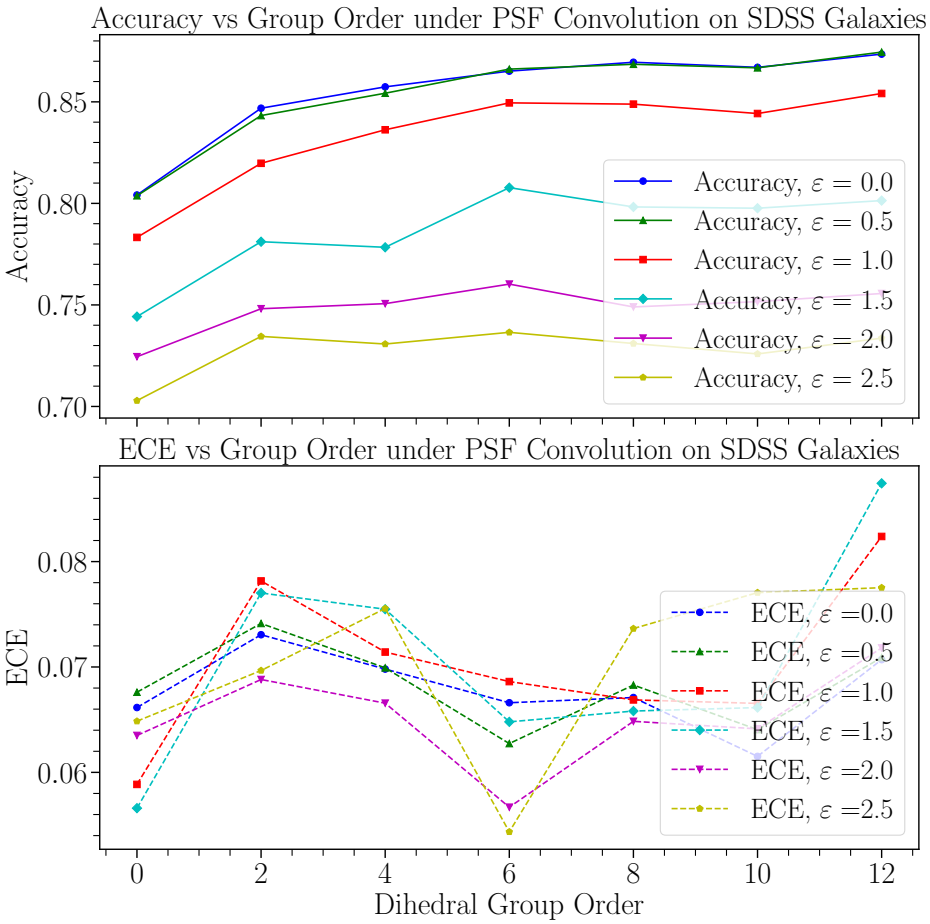



Figure 52: Accuracy and ECE vs Dihedral Group Order under PSF Convolution on SDSS Galaxies.

# I Additional Spectra Results

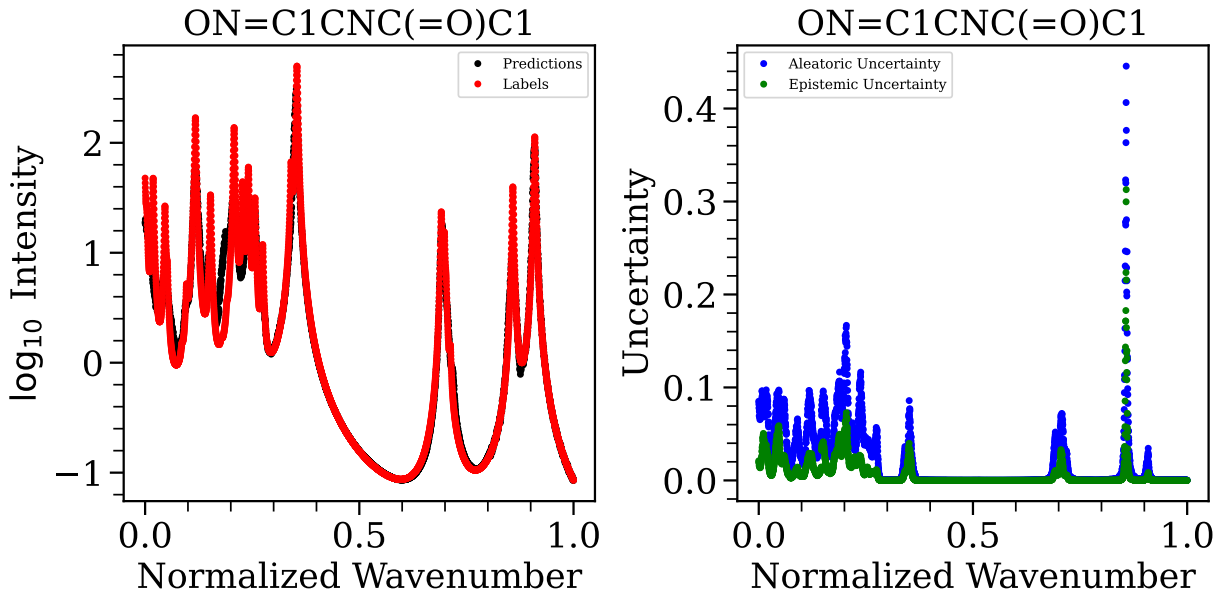



Figure 53: Same as Figure 22 for the molecule given by SMILES string $ON = C1CNC(= O)C1$.

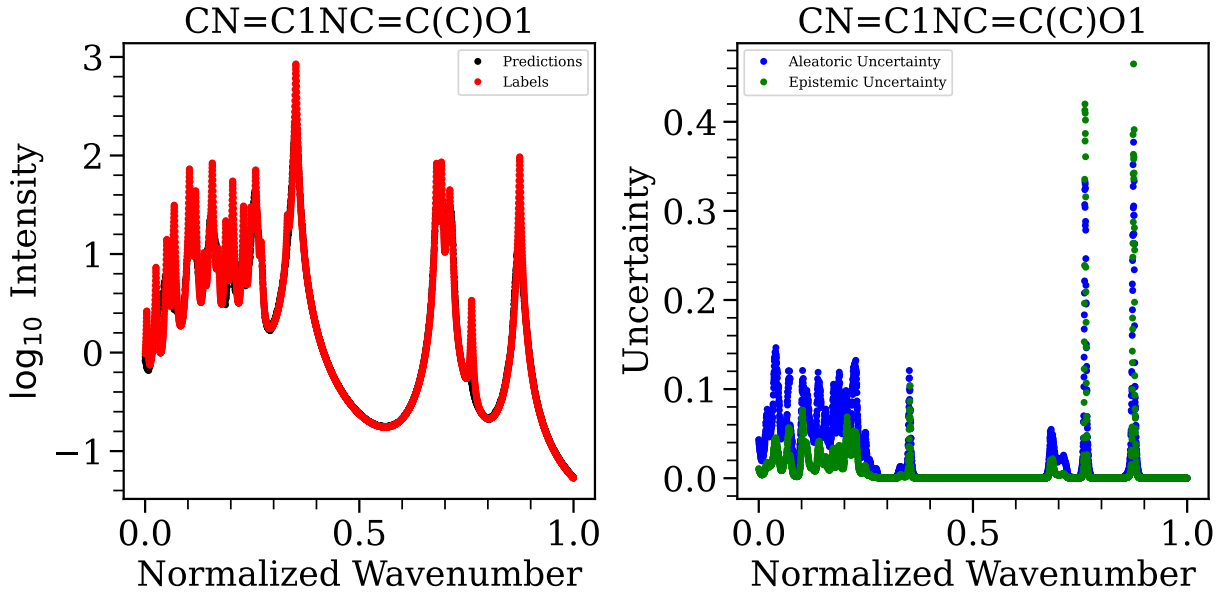



Figure 54: Same as Figure 22 for the molecule given by SMILES string $CN = C1NC = C(C)O1$.

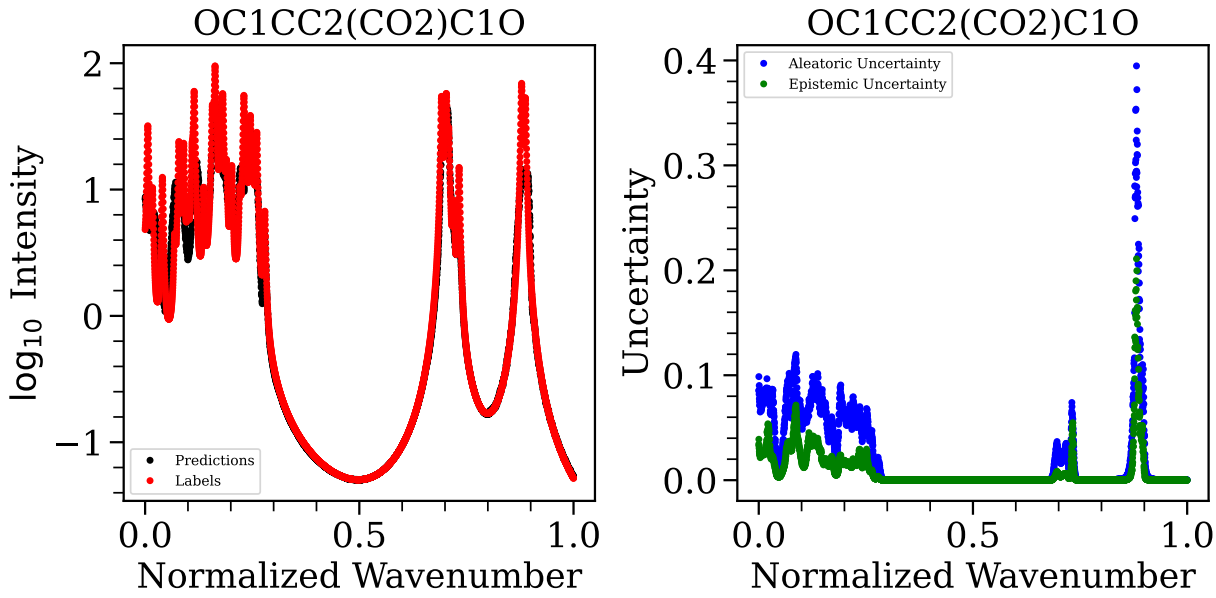



Figure 55: Same as Figure 22 for the molecule given by SMILES string $OC1CC2(CO2)C10$.

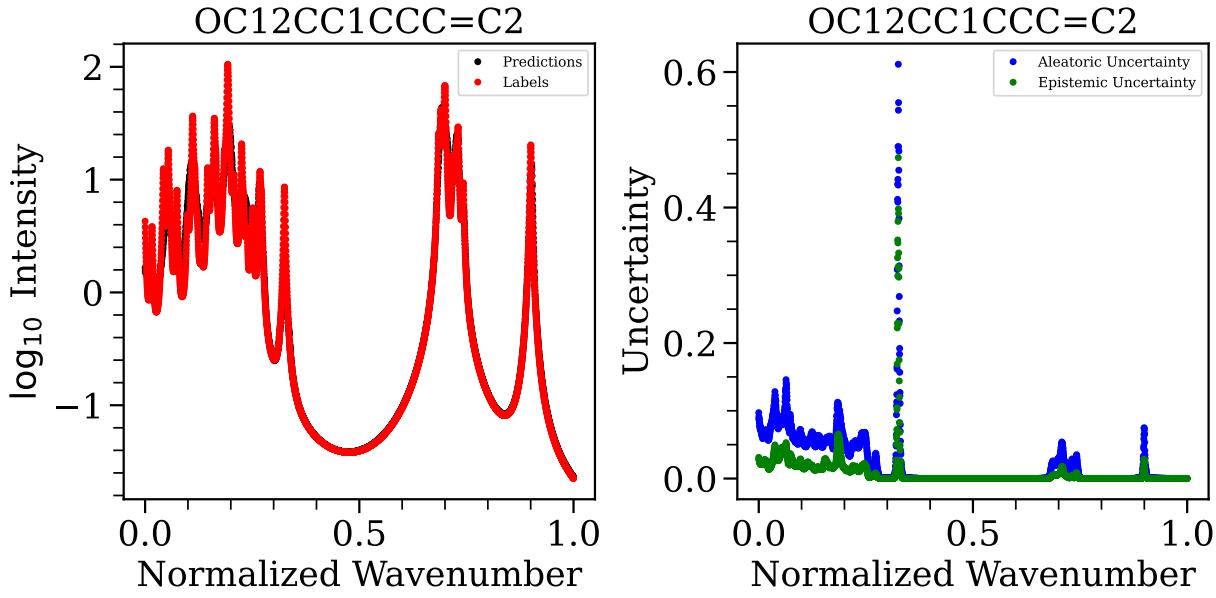



Figure 56: Same as Figure 22 for the molecule given by SMILES string $OC12CC1CCC=C2$.

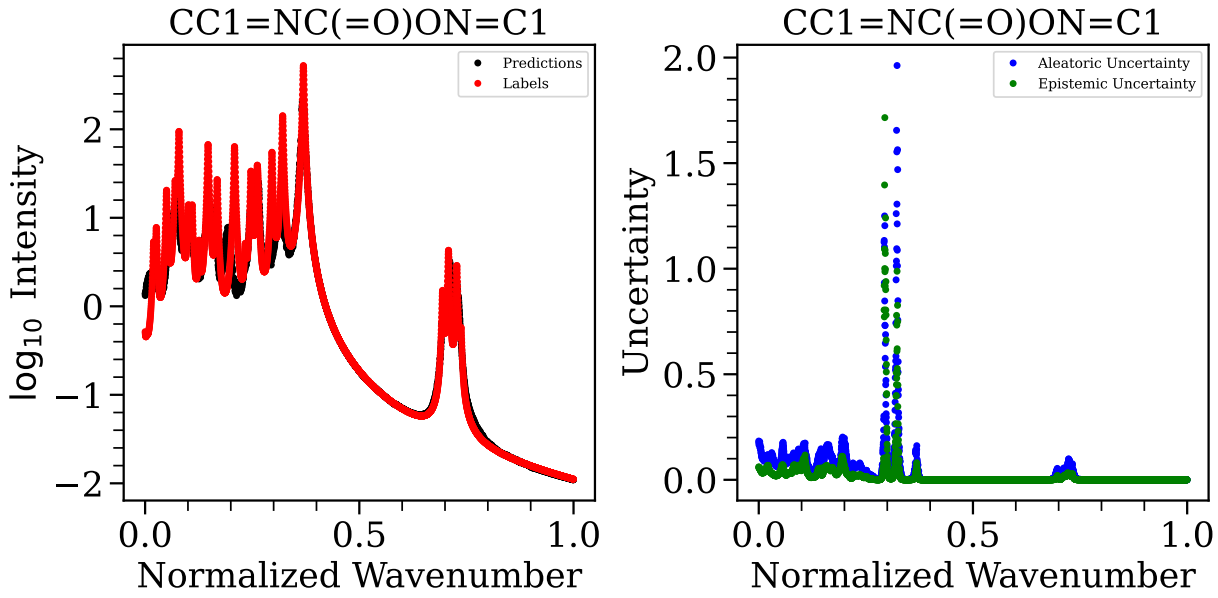



Figure 57: Same as Figure 22 for the molecule given by SMILES string $CC1 = NC(= O)ON = C1$.

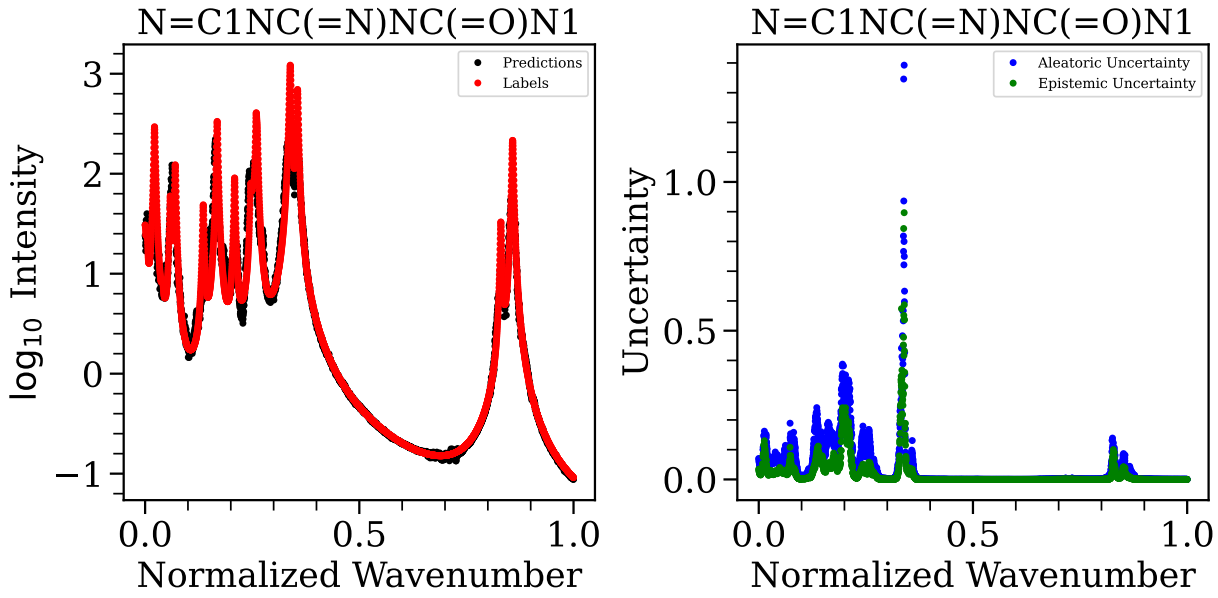



Figure 58: Same as Figure 22 for the molecule given by SMILES string $N = C1NC(= N)NC(= O)N1$.

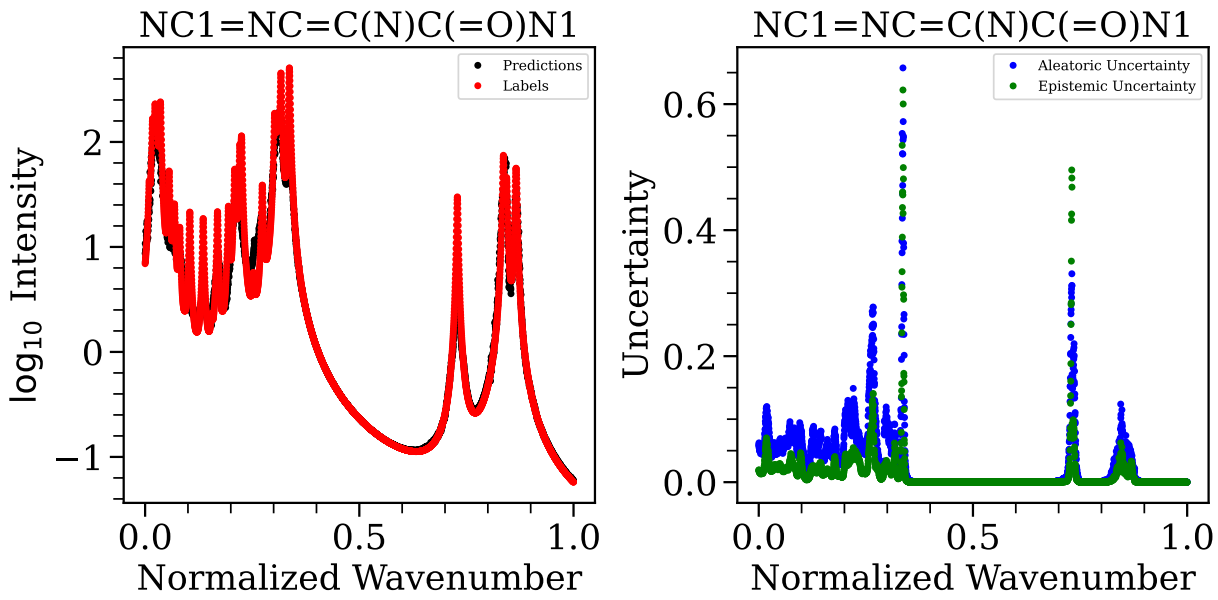



Figure 59: Same as Figure 22 for the molecule given by SMILES string $NC1 = NC = C(N)C(= O)N1$.

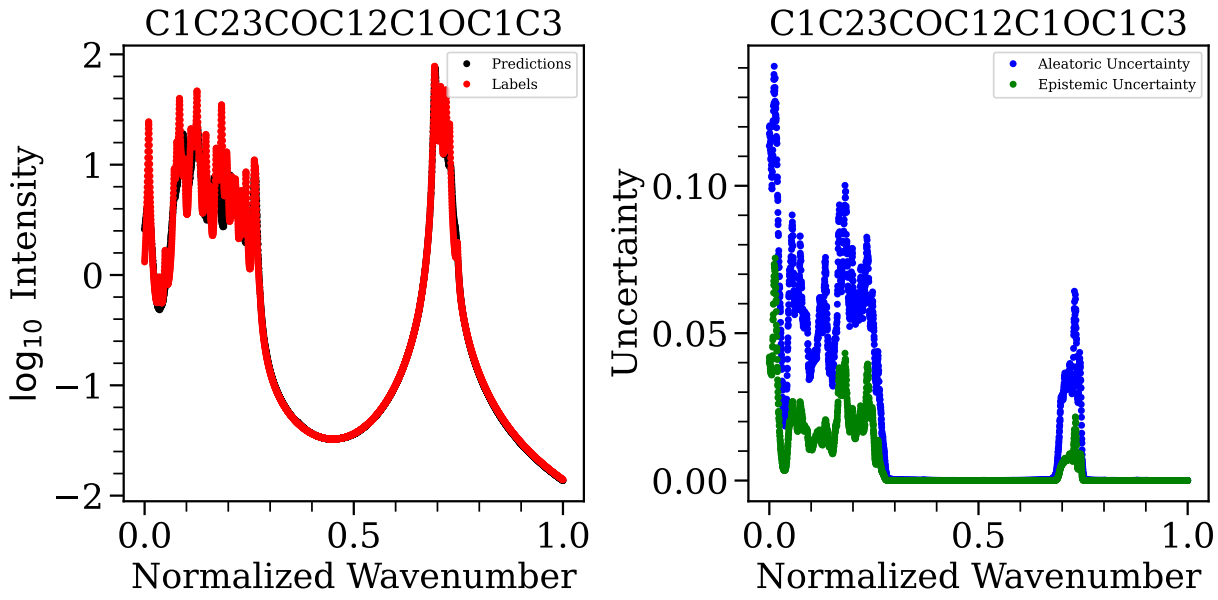



Figure 60: Same as Figure 22 for the molecule given by SMILES string $C1C23COC12C1OC1C3$.

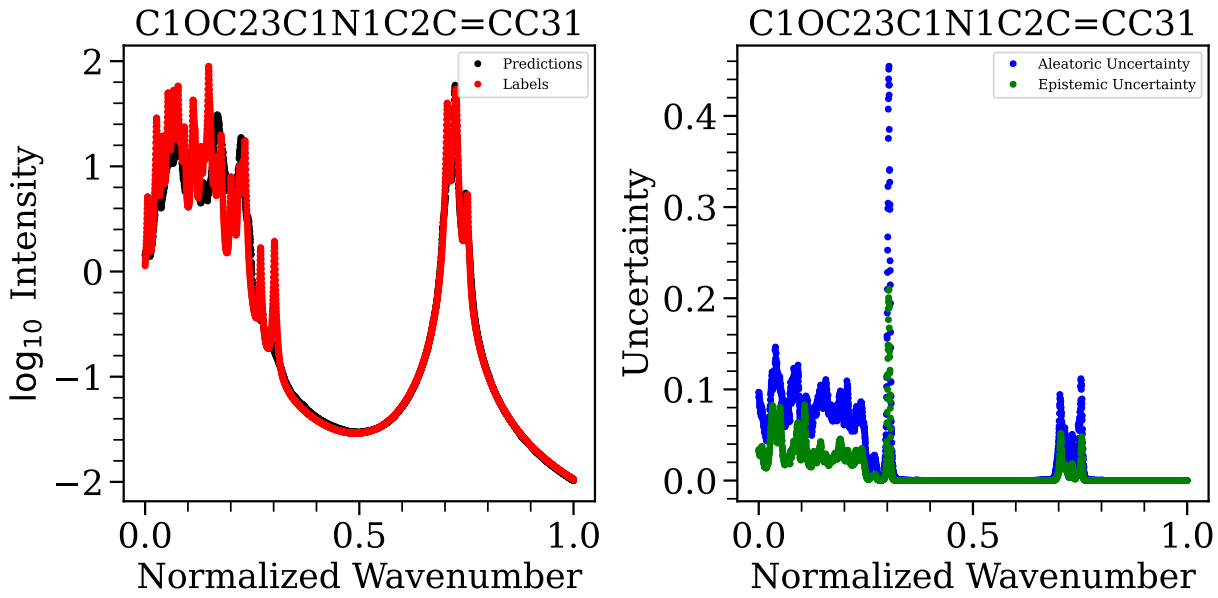



Figure 61: Same as Figure 22 for the molecule given by SMILES string $C1OC23C1N1C2C = CC31$.

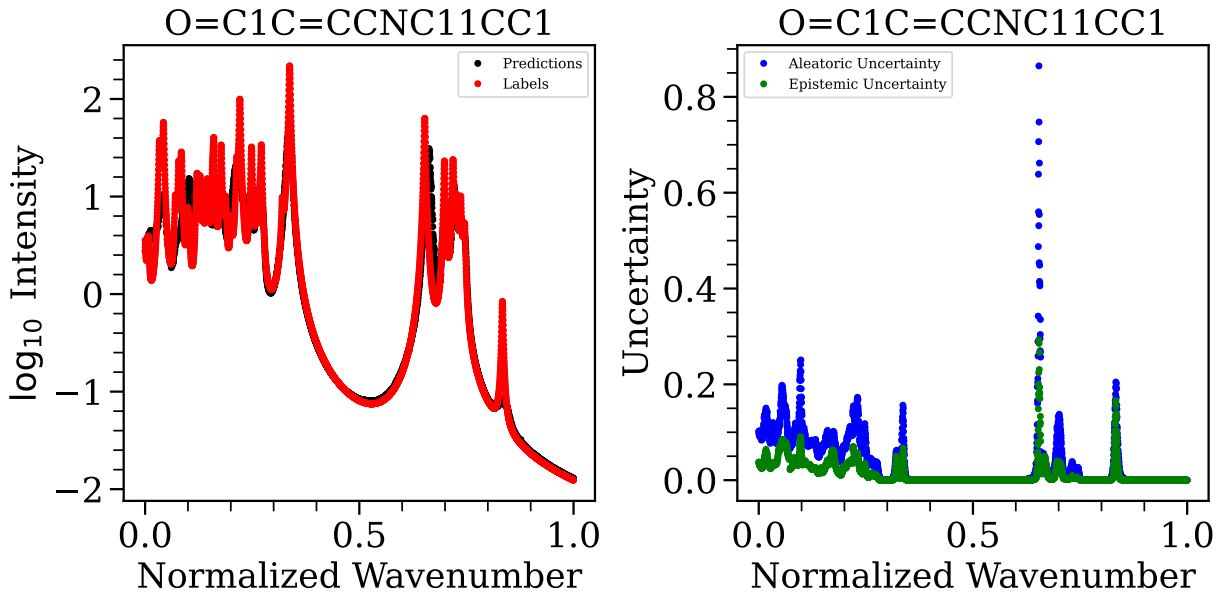



Figure 62: Same as Figure 22 for the molecule given by SMILES string $O = C1C = CCNC11CC1$.

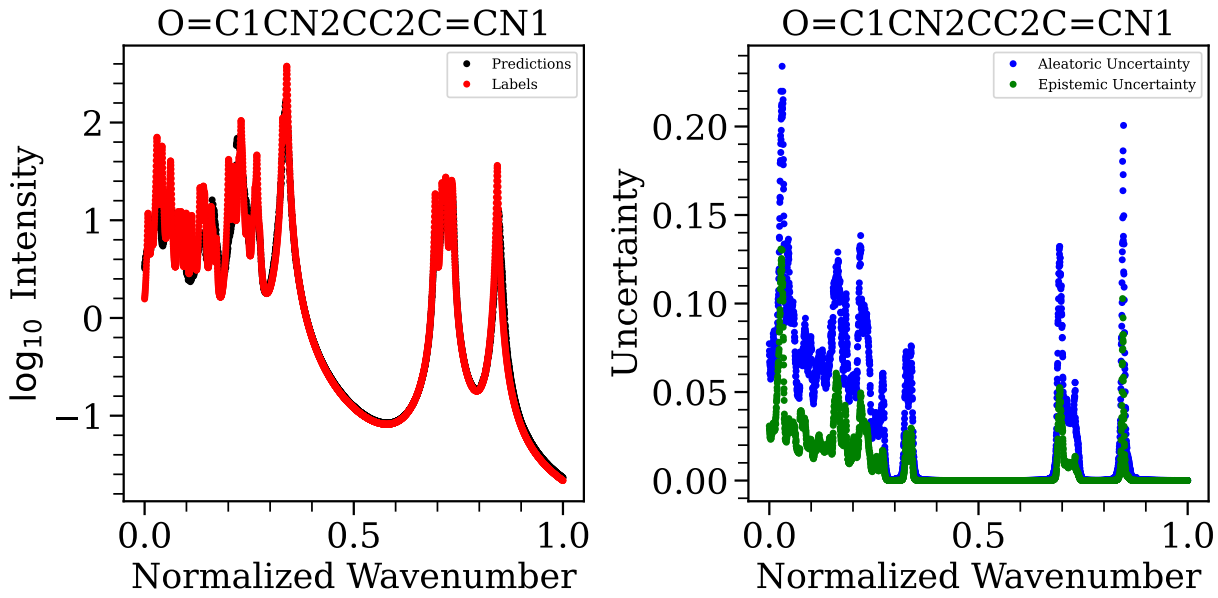



Figure 63: Same as Figure 22 for the molecule given by SMILES string $O = C1CN2CC2C = CN1$.

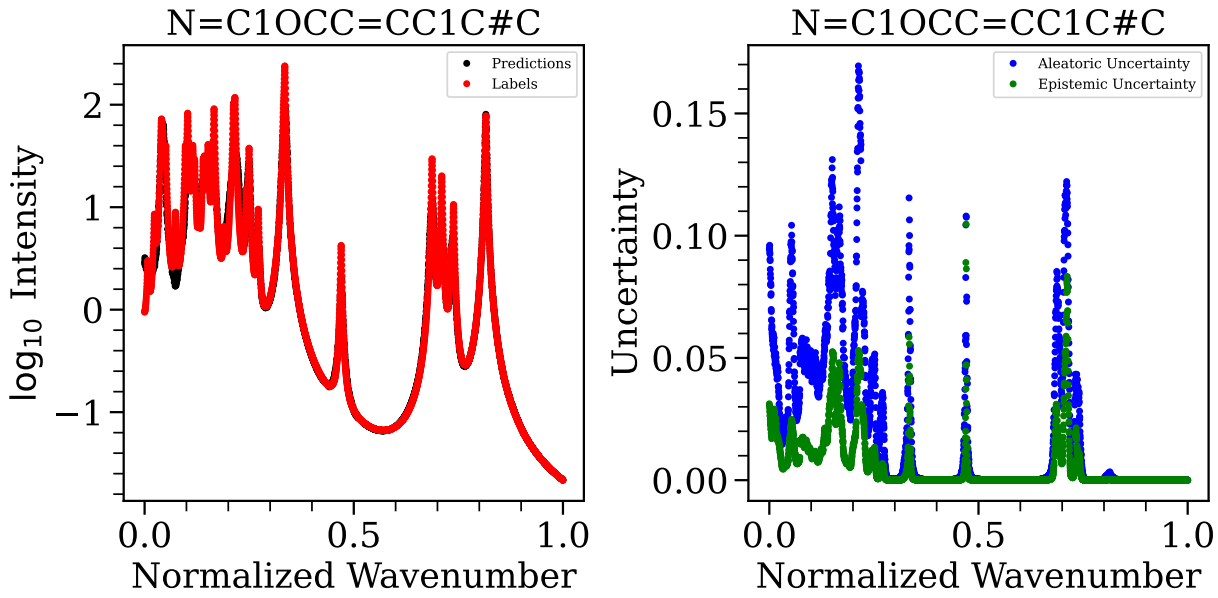



Figure 64: Same as Figure 22 for the molecule given by SMILES string $N = C1OCC = CC1C\#C$.

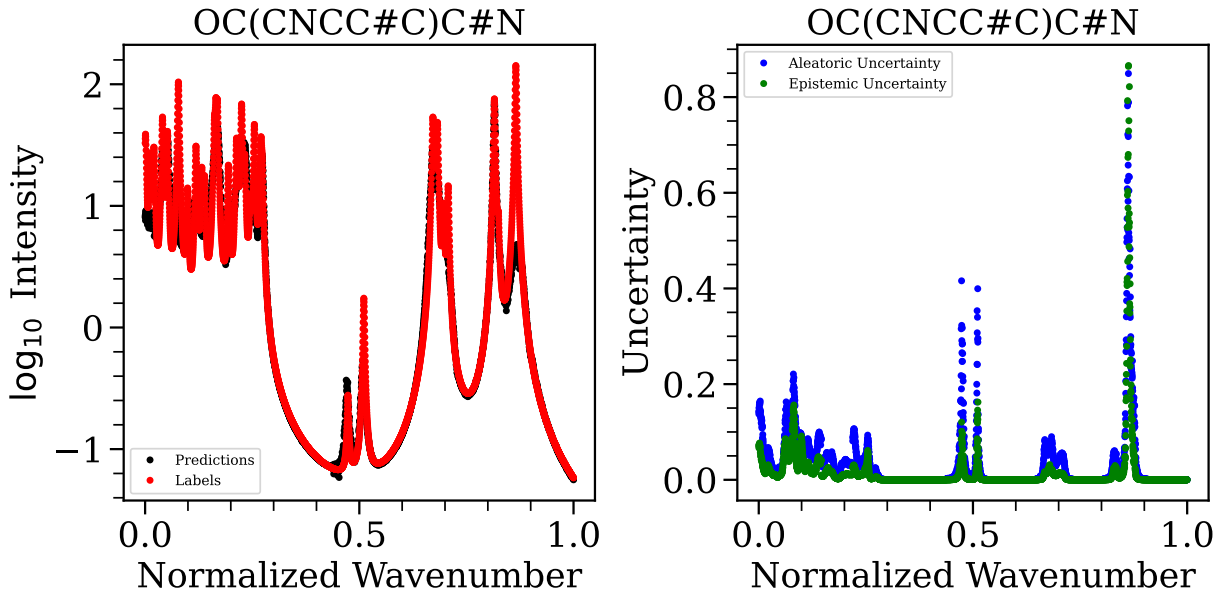



Figure 65: Same as Figure 22 for the molecule given by SMILES string $OC(CNCC\#C)C\#N$.

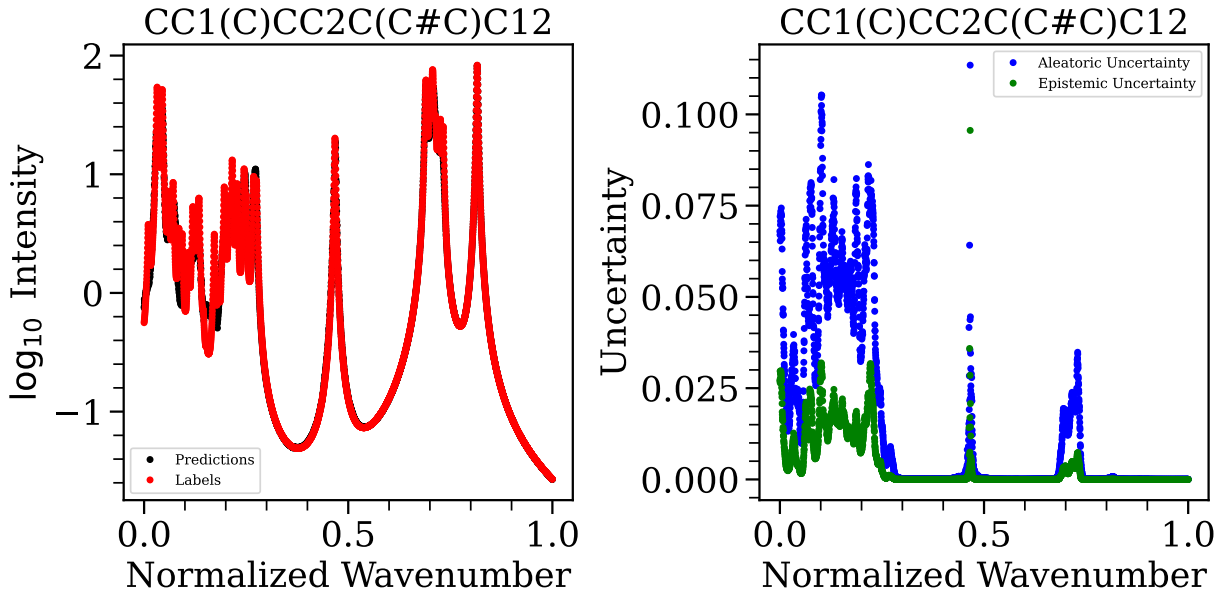



Figure 66: Same as Figure 22 for the molecule given by SMILES string $CC1(C)CC2C(C\#C)C12$.

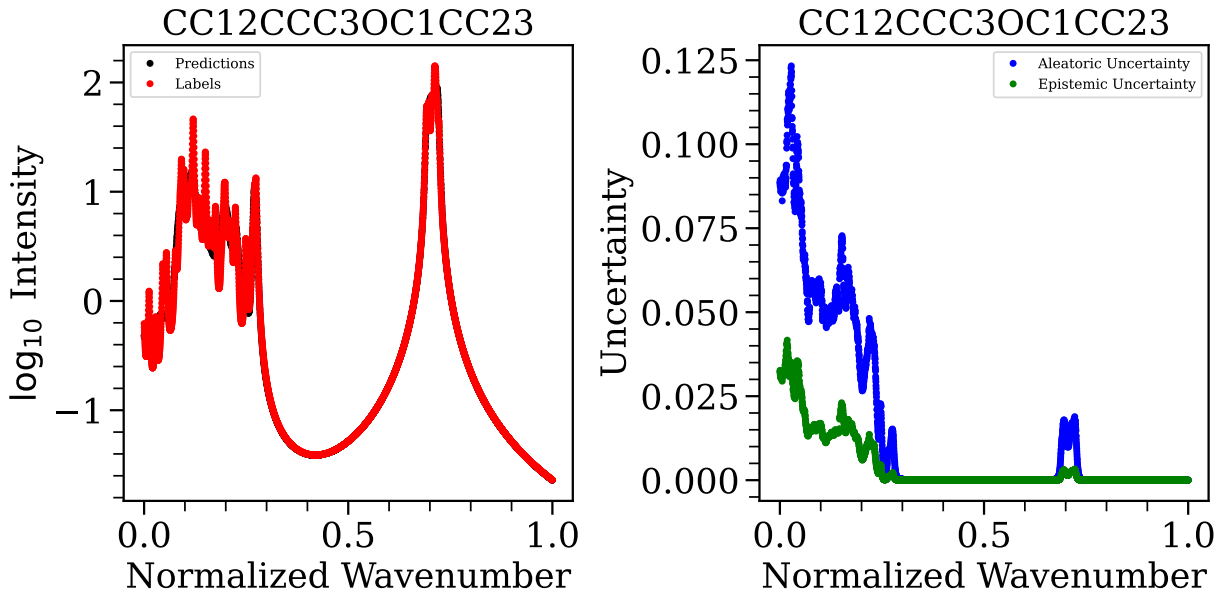



Figure 67: Same as Figure 22 for the molecule given by SMILES string $CC12CCC3OC1CC23$.

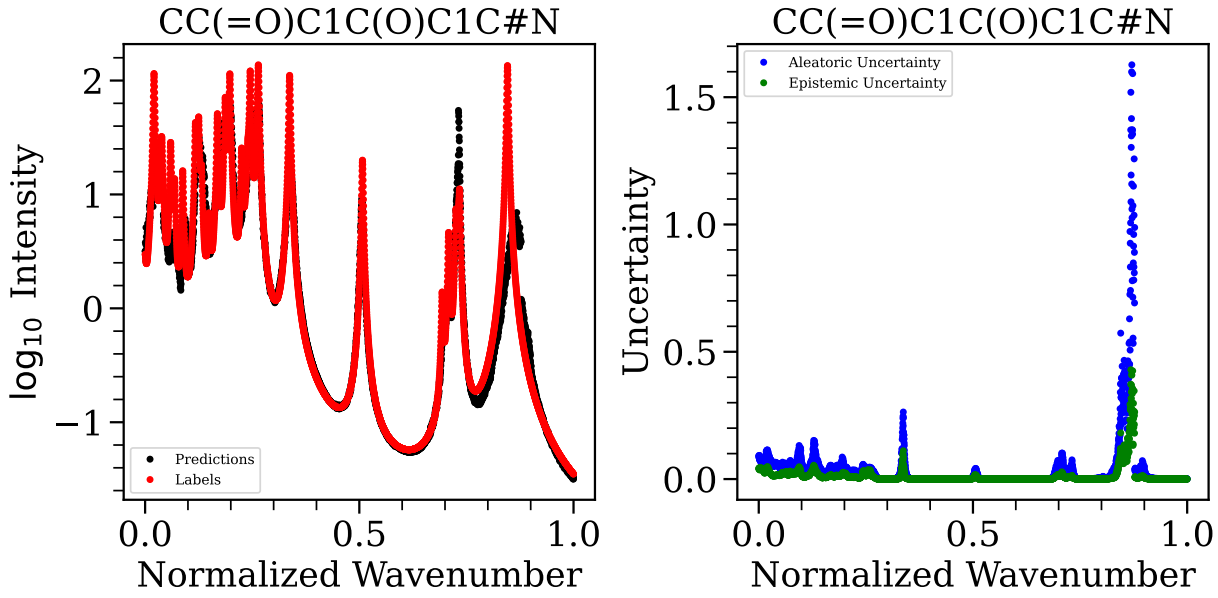



Figure 68: Same as Figure 22 for the molecule given by SMILES string $CC(=O)C1C(O)C1C\#N$.

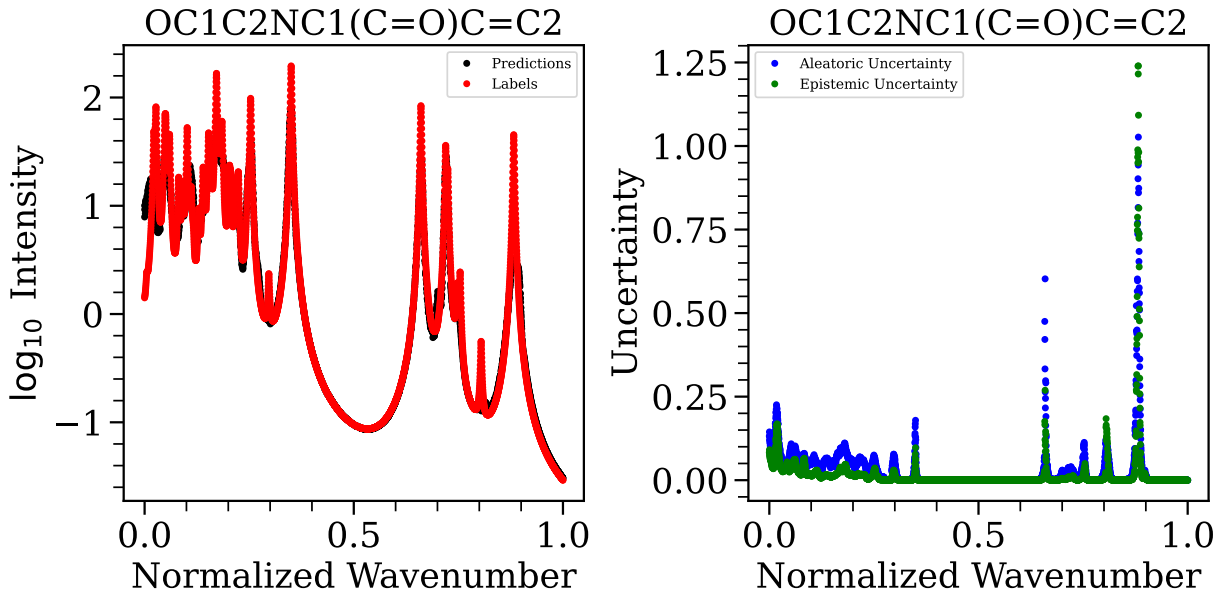



Figure 69: Same as Figure 22 for the molecule given by SMILES string $OC1C2NC1(C=O)C=C2$.

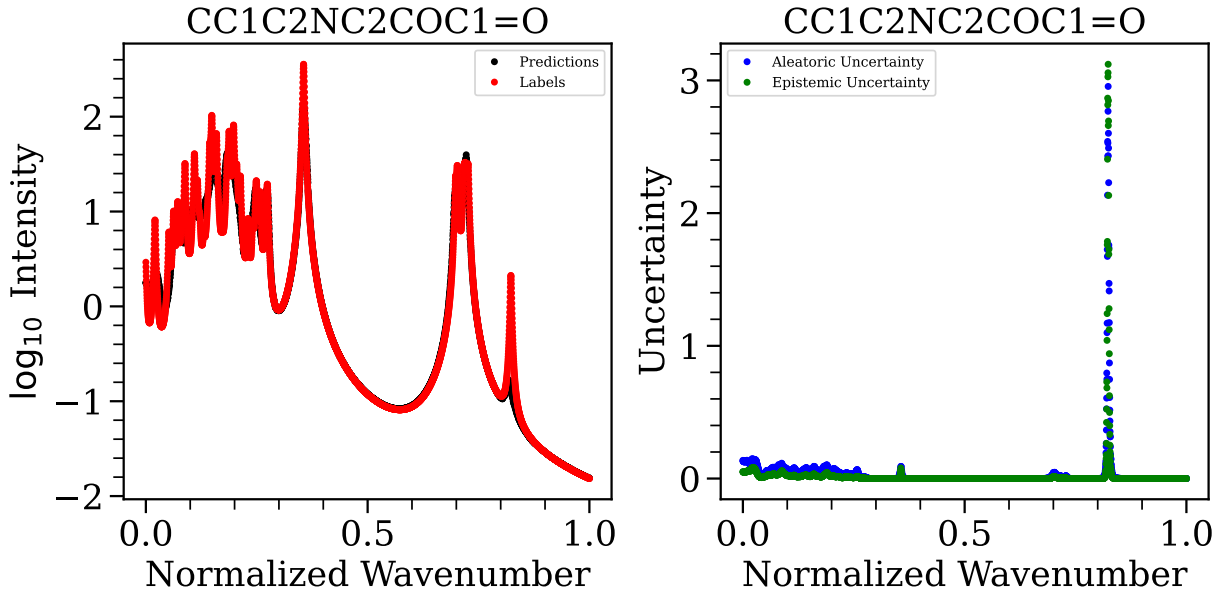



Figure 70: Same as Figure 22 for the molecule given by SMILES string $CC1C2NC2COC1=O$.

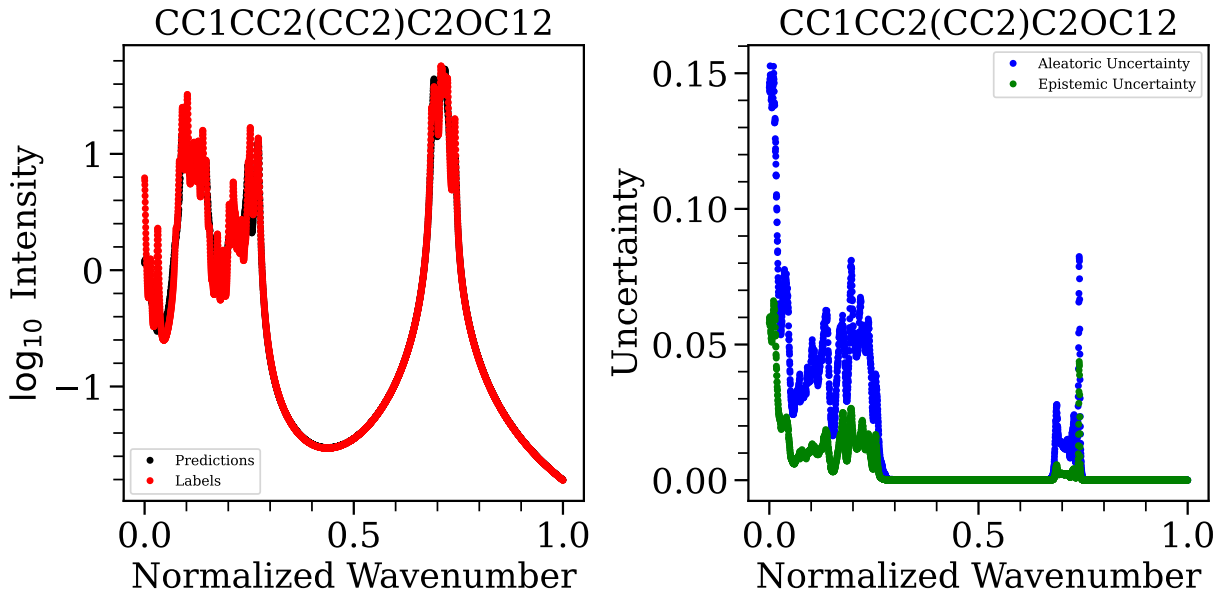



Figure 71: Same as Figure 22 for the molecule given by SMILES string $CC1CC2(CC2)C2OC12$.

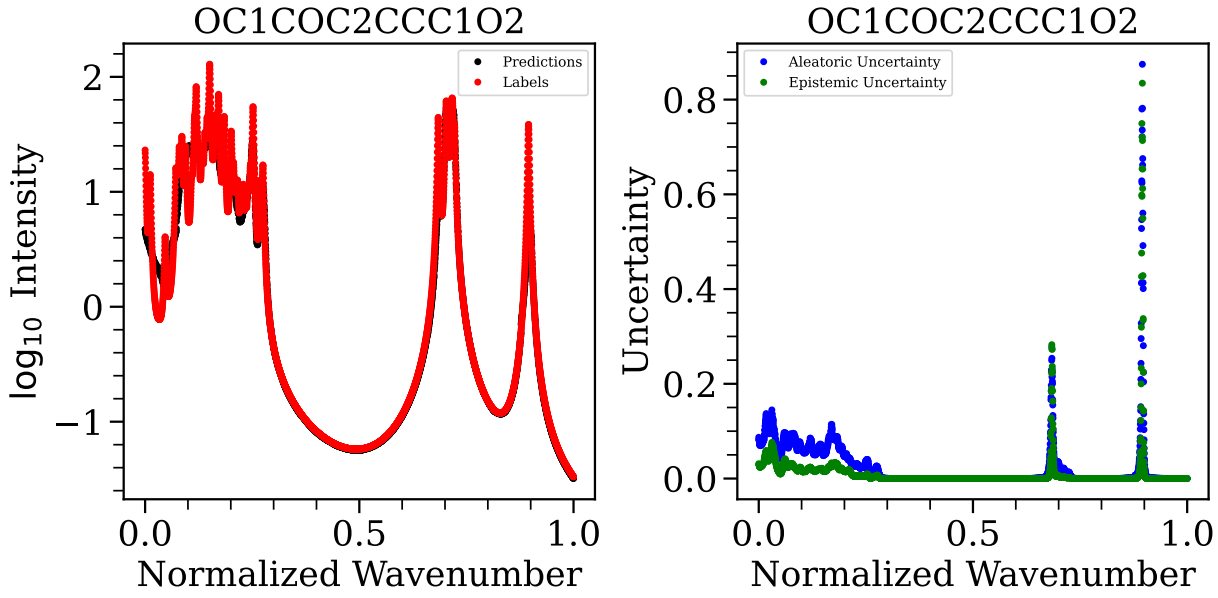



Figure 72: Same as Figure 22 for the molecule given by SMILES string $OC1COC2CCC1O2$.

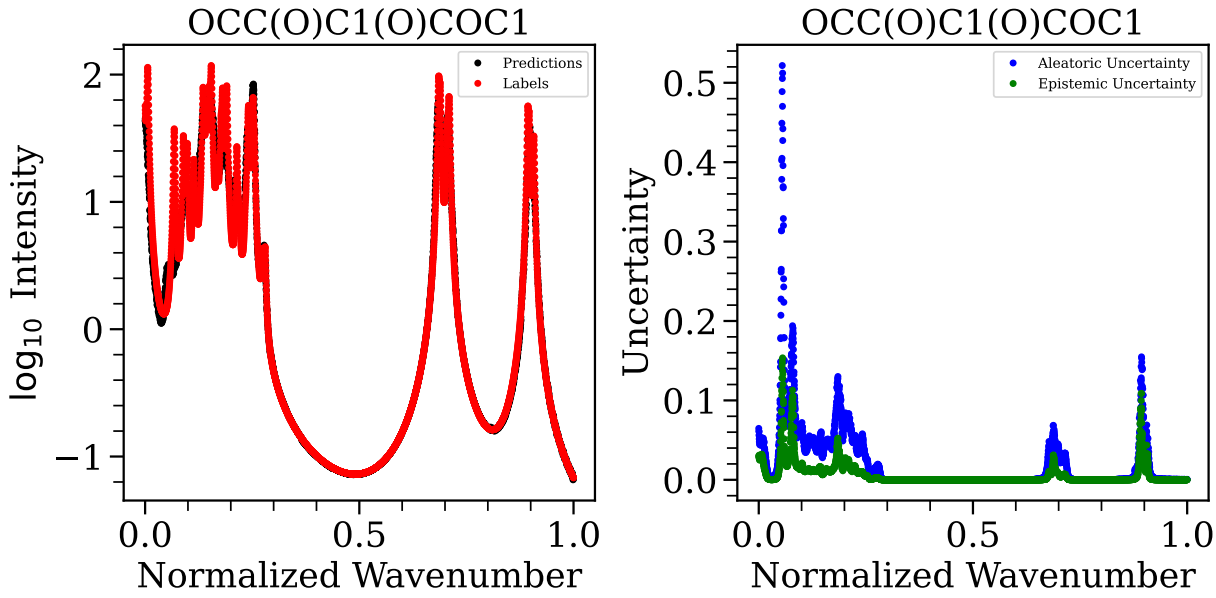



Figure 73: Same as Figure 22 for the molecule given by SMILES string $OCC(O)C1(O)COC1$.

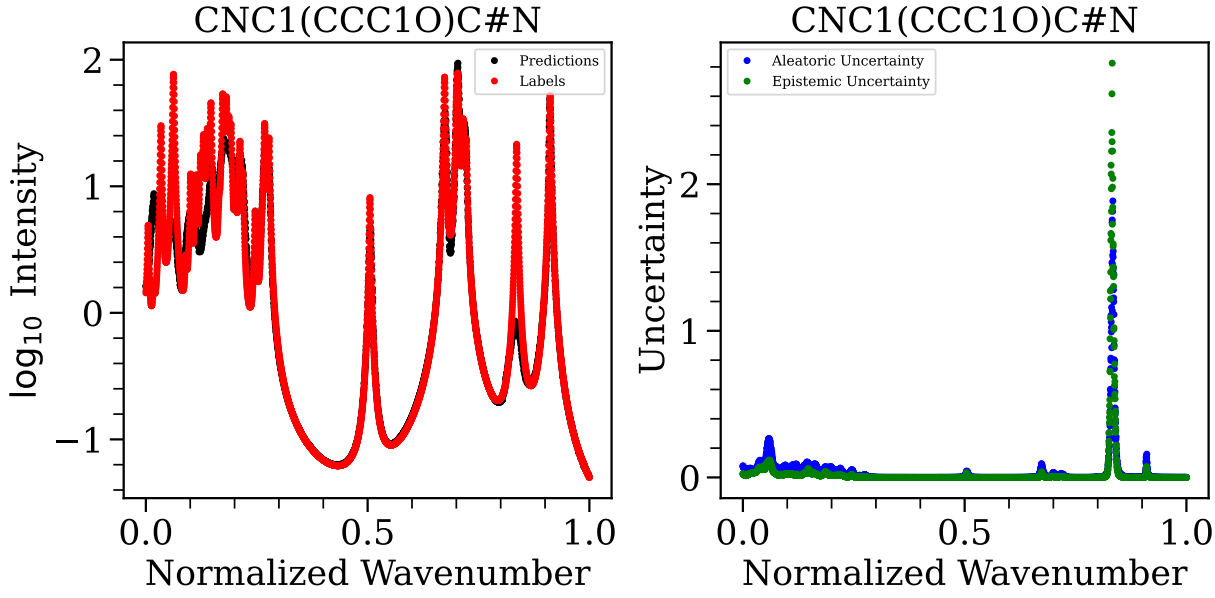



Figure 74: Same as Figure 22 for the molecule given by SMILES string $CNC1(CCC1O)C\#N$.

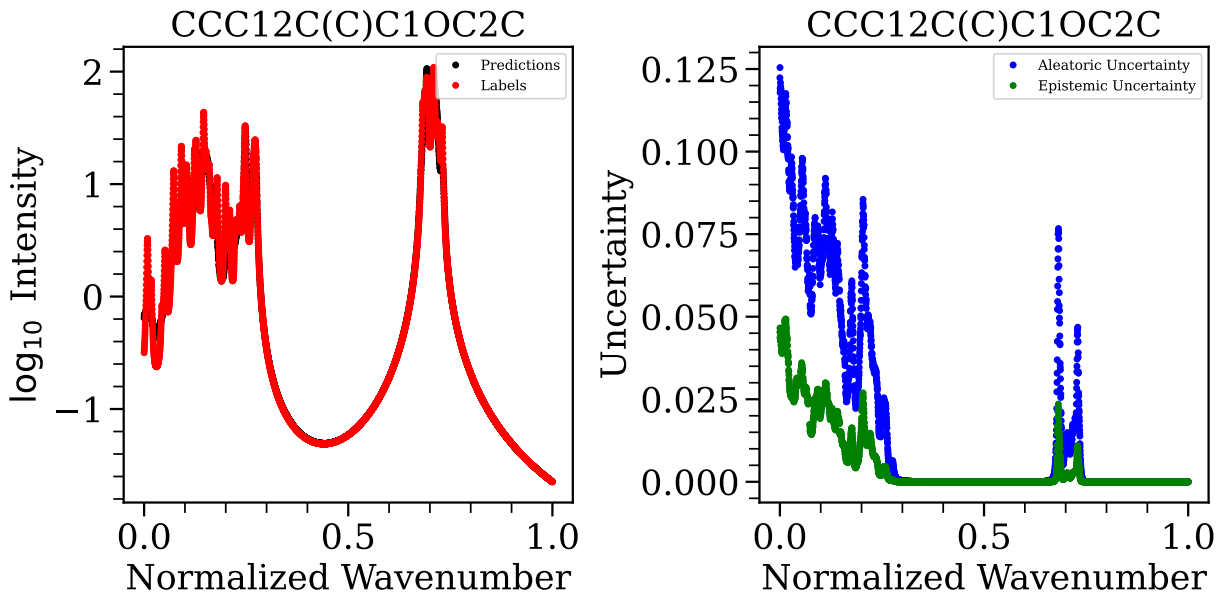



Figure 75: Same as Figure 22 for the molecule given by SMILES string $CCC12C(C)C1OC2C$.

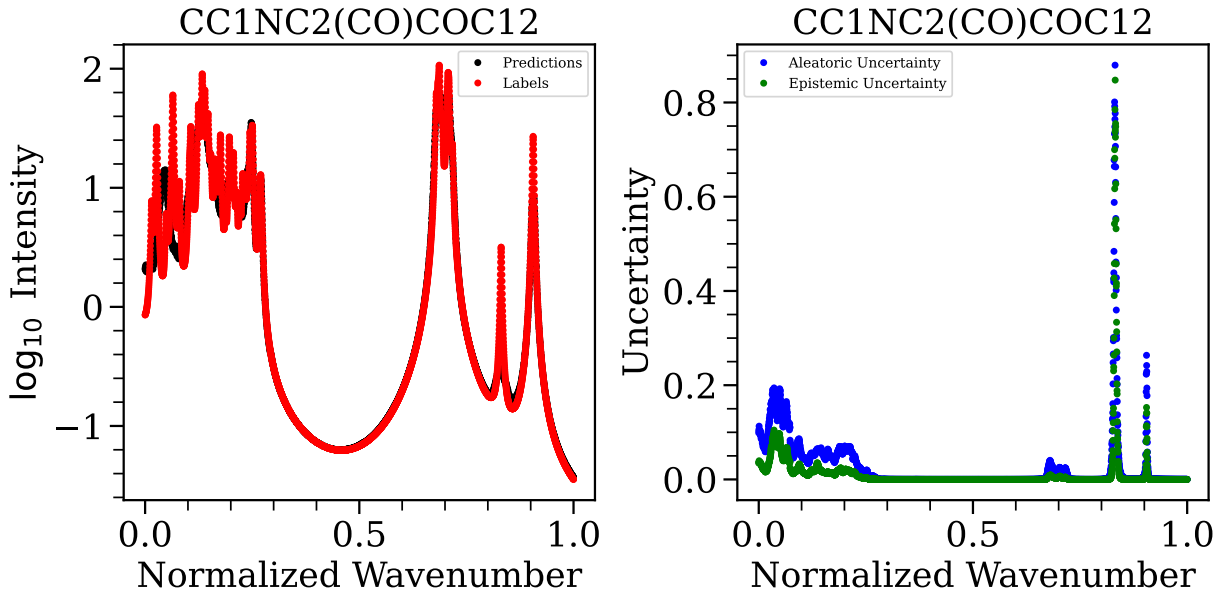



Figure 76: Same as Figure 22 for the molecule given by SMILES string $CC1NC2(CO)COC12$.

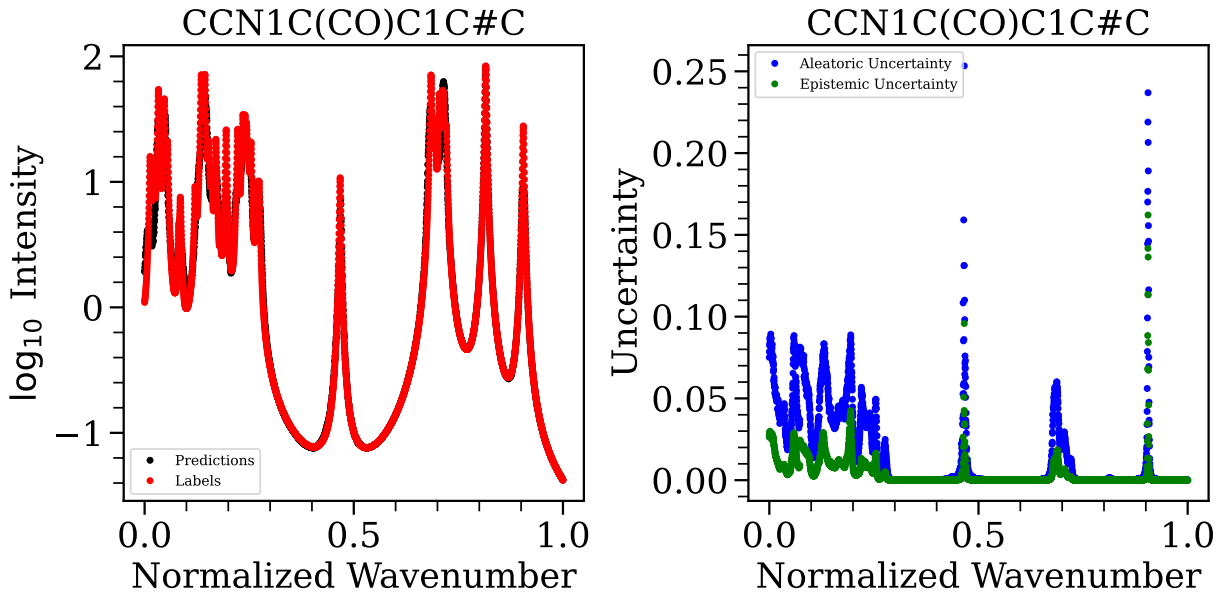



Figure 77: Same as Figure 22 for the molecule given by SMILES string $CCN1C(CO)C1C\#C$.

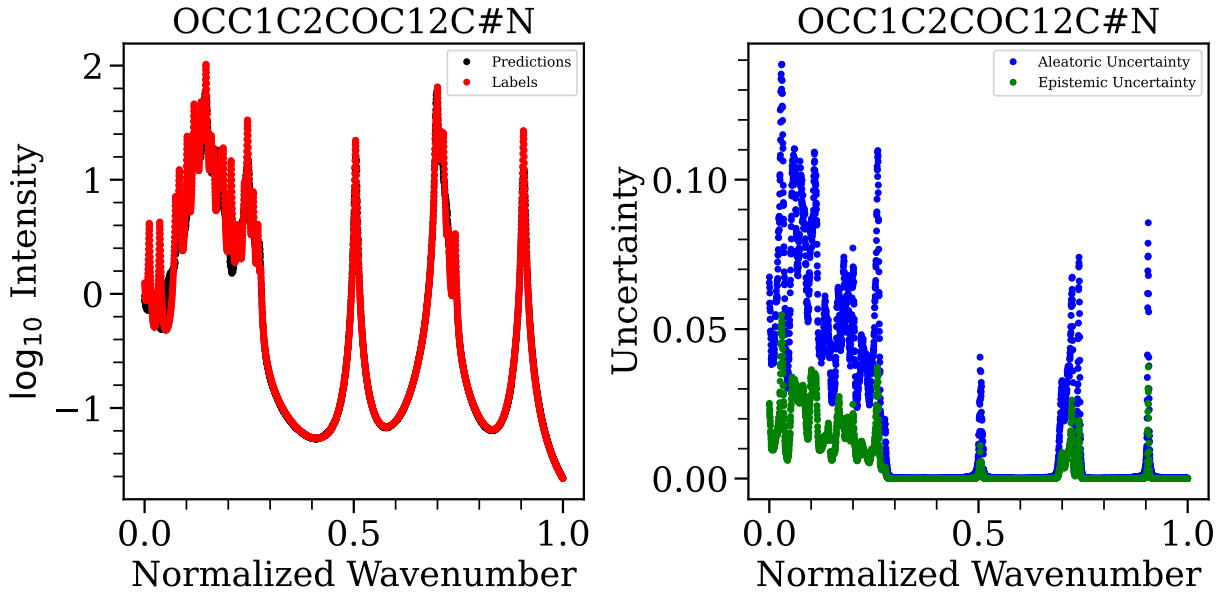



Figure 78: Same as Figure 22 for the molecule given by SMILES string $OCC1C2COC12C\#N$.

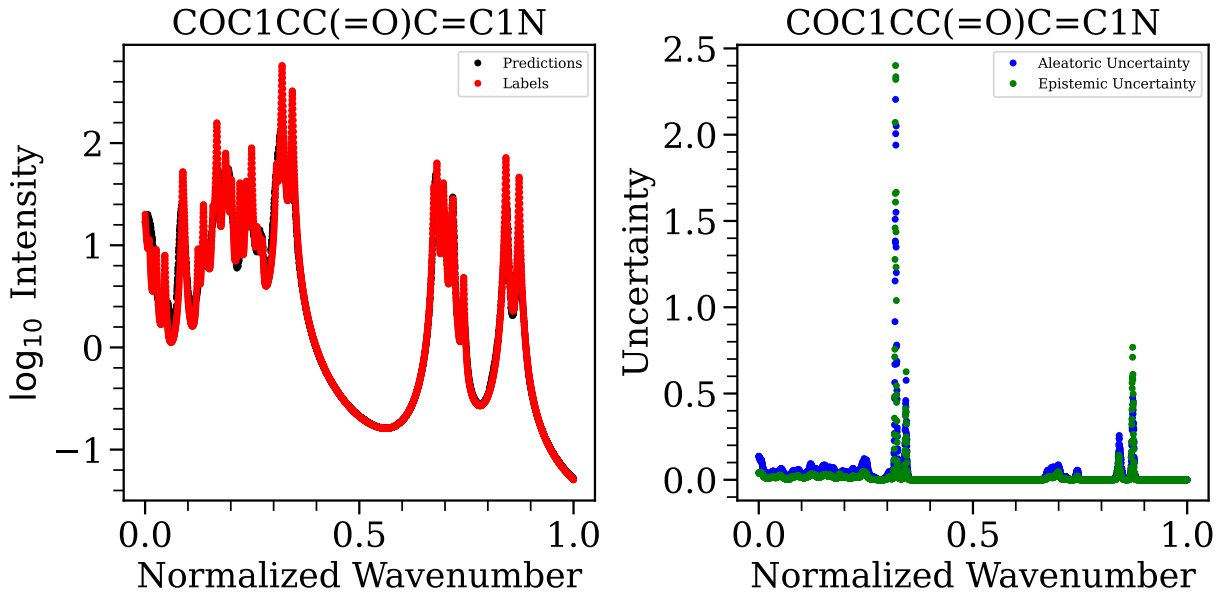



Figure 79: Same as Figure 22 for the molecule given by SMILES string $COC1CC(=O)C=C1N$.

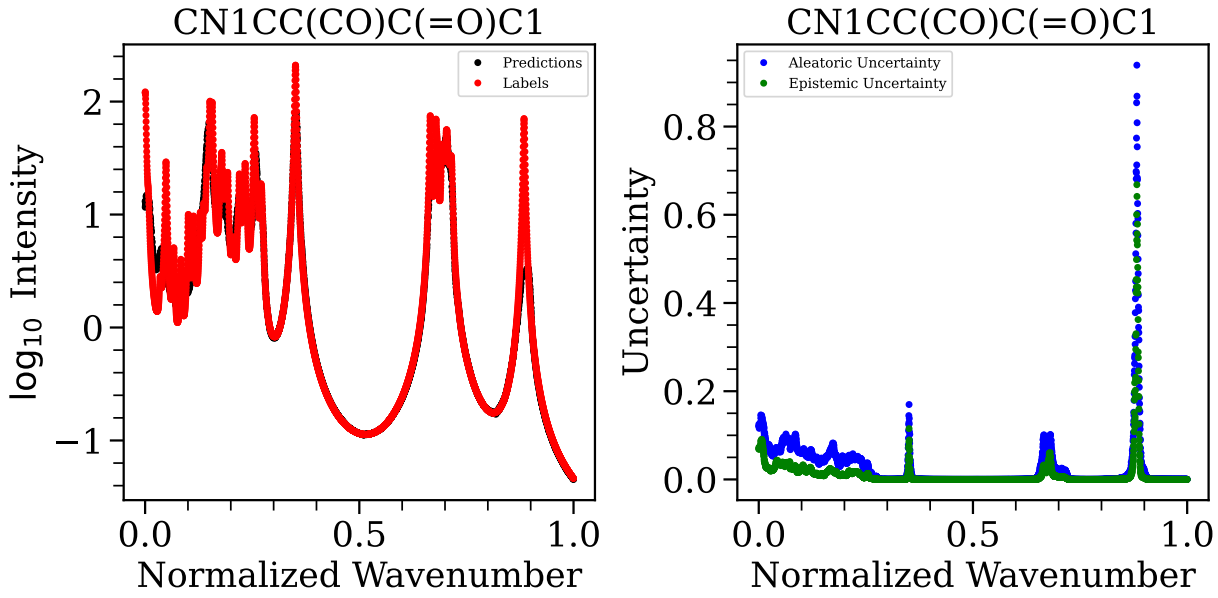



Figure 80: Same as Figure 22 for the molecule given by SMILES string $CN1CC(CO)C(=O)C1$.

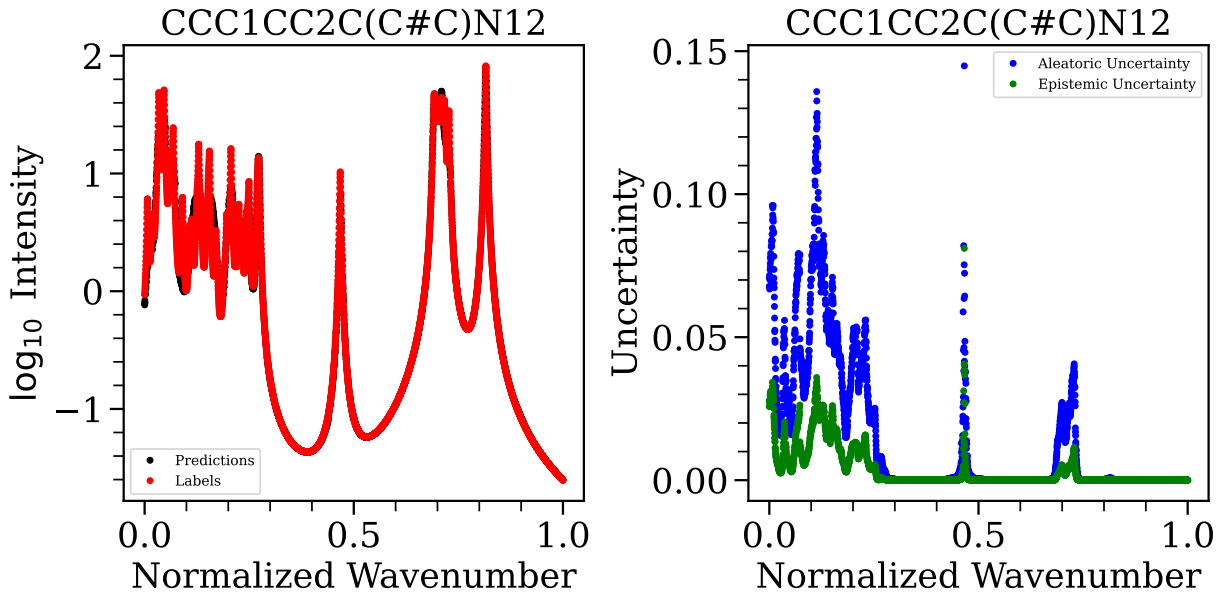



Figure 81: Same as Figure 22 for the molecule given by SMILES string $CCC1CC2C(C\#C)N12$.

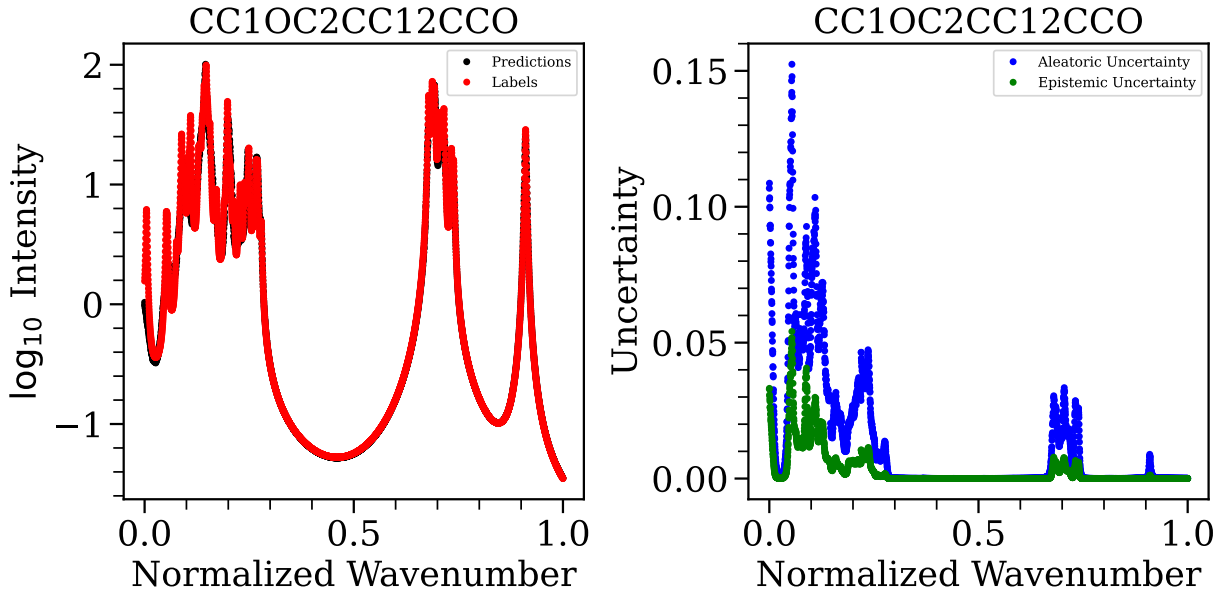



Figure 82: Same as Figure 22 for the molecule given by SMILES string $CC1OC2CC12CCO$.

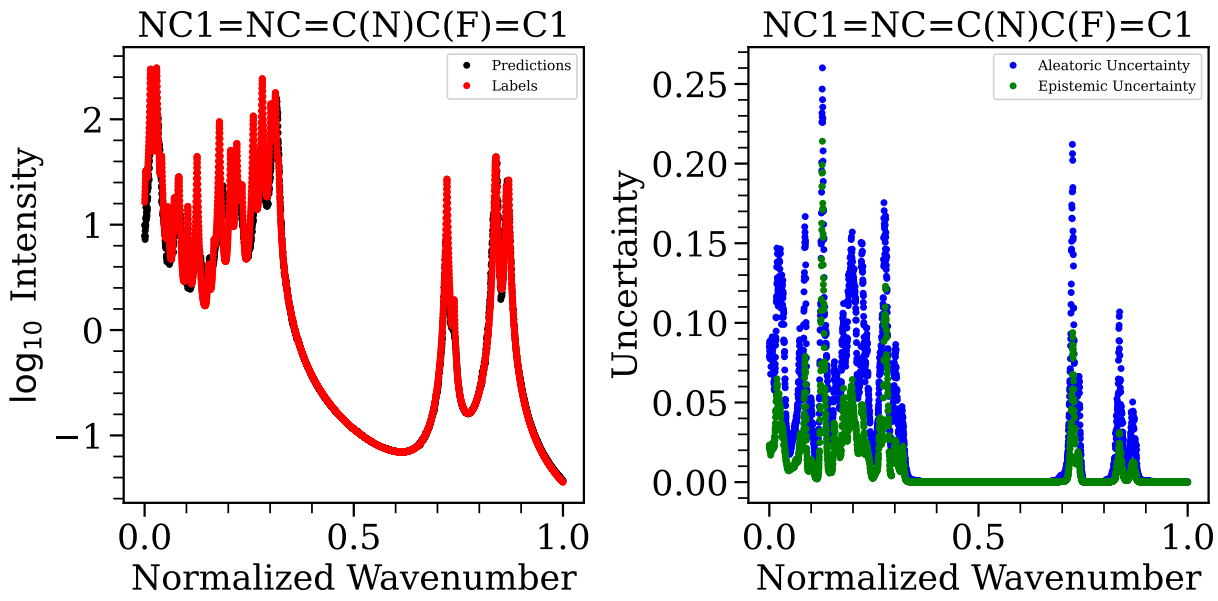



Figure 83: Same as Figure 22 for the molecule given by SMILES string $NC1 = NC = C(N)C(F) = C1$.

## J Relation to Classification Error

If we start with

$$c \leq \int_{p=0}^{p=1} r(p)\text{MAX}(p)dp \tag{132}$$

$$= \int_{p=0}^{p=1} r(p)\mathbb{P}\left(f(x) = h_Y(x) | h_P(x) = p\right) dp \tag{133}$$

$$= \mathbb{P}\left(f(x) = h_Y(x)\right) \tag{134}$$

Guo et al. (2017) tells us that accuracy, given by Equation 120, is an unbiased estimator for $\mathbb{P}\left(f(x) = h_Y(x)\right)$ (as long as we let $m = 1$ such that $I_m$ is the entire domain). Notice that Equation 120 is reminiscent of Equation 26. Indeed, the expected value for an indicator function is just the probability. Thus, we may write that

$$\mathbb{P}\left(f(x) = h_Y(x)\right) = \mathbb{E}_{x\sim q}\left[\mathbb{I}(f(x) = h_Y(x))\right] \tag{135}$$

$$= \mathbb{E}_{x\sim q}\left[1 - \mathbb{I}(f(x) \neq h_Y(x))\right] \tag{136}$$

$$= \mathbb{E}_{x\sim q}\left[1\right] - \mathbb{E}_{x\sim q}\left[\mathbb{I}(f(x) \neq h_Y(x))\right] \tag{137}$$

$$= 1 - \text{err}_{\text{cls}}(h). \tag{138}$$

This relationship is useful for comparison with Wang et al. (2024) and provides an alternative way to intuit the accuracy of a correct label prediction.

# Part X

# Part V Appendix

## A Additional PSF Diagnostic Figures

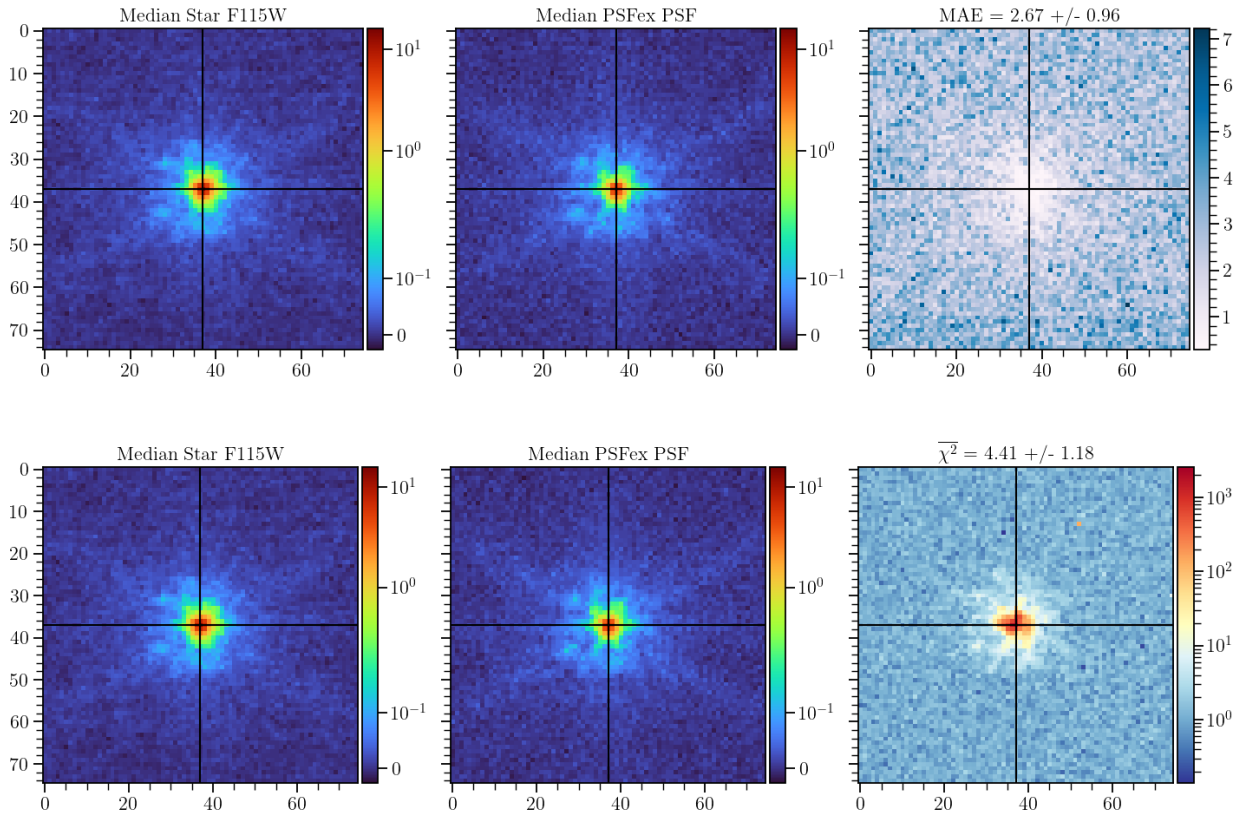



Figure 84: Examples of mean average error (top panel) and $\chi^2$ residual figures (bottom panel), shown here for the simulated mosaics in F115W.

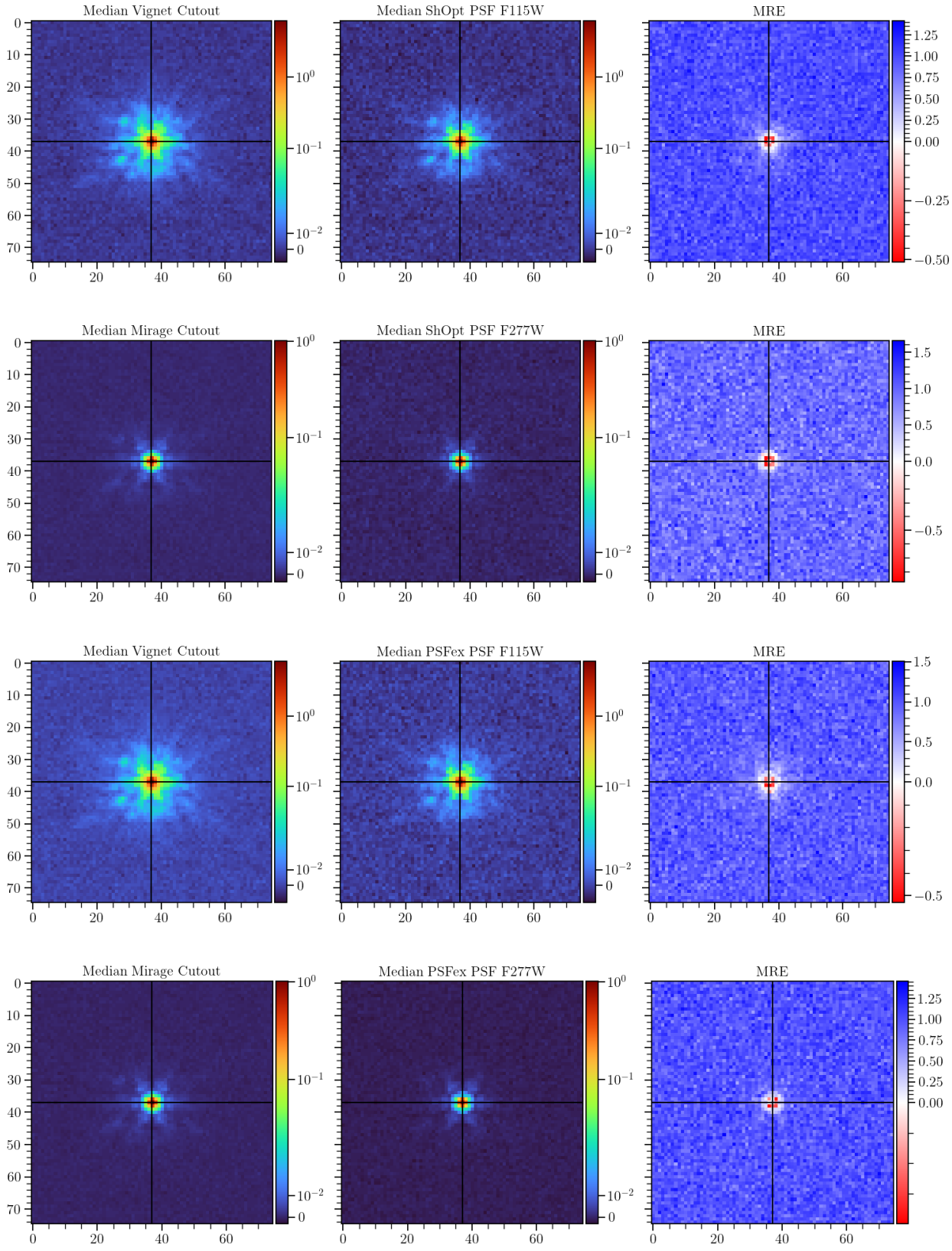



Figure 85: Mean relative error between MIRAGE input point source images and PSF models for the F115W and F150W bandpasses. Panels and color bars are the same as Figure 34.

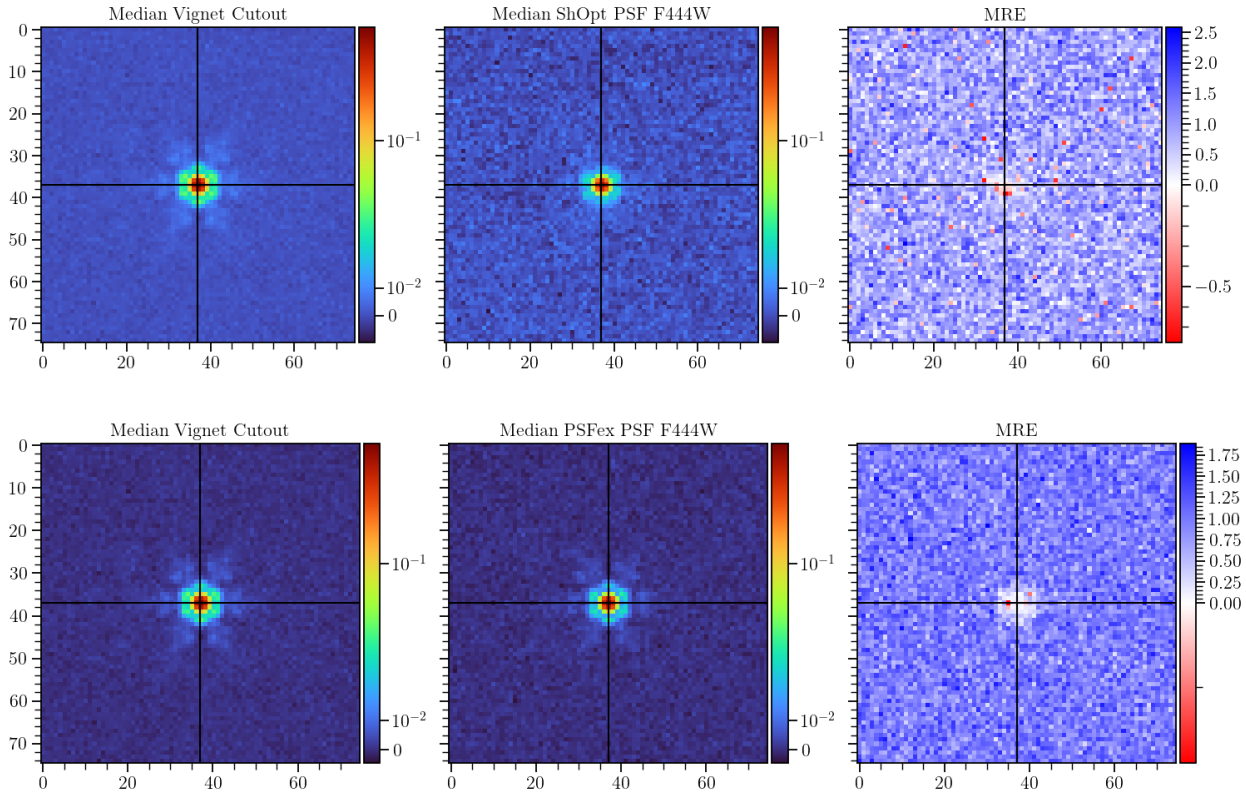



Figure 86: Mean relative error between MIRAGE input point source images and PSF models for the F444W filter. Panels and color bars are the same as Figure 34.

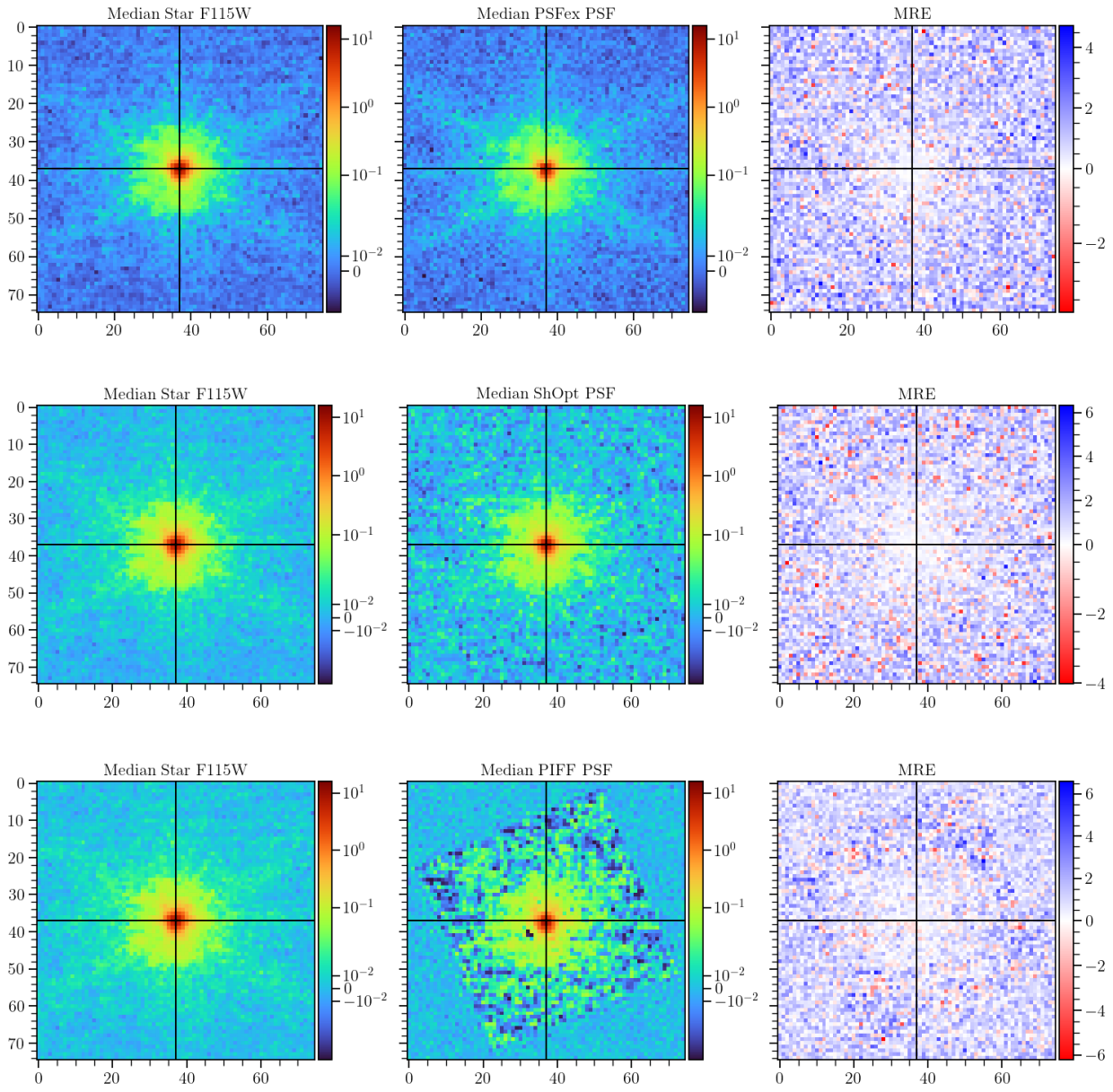



Figure 87: Mean relative error between stars and PSF models for simulated mosaics in the F115W bandpass. Left panels show the median of the vignettes. Panels and color bars are the same as in Figure 36.

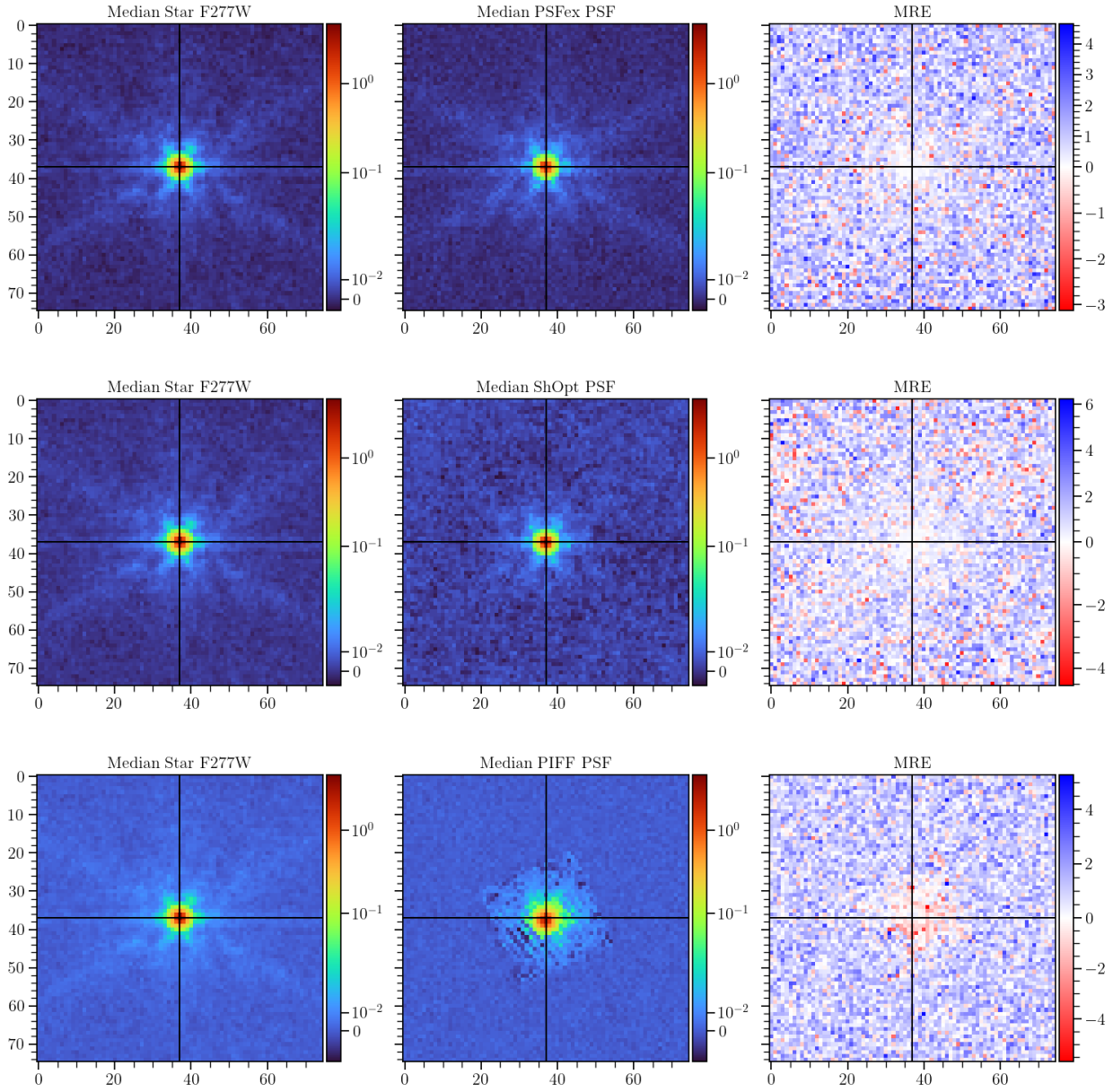



Figure 88: Evaluation of mean relative error between stars and PSF models for simulated mosaics in the F277W bandpass. Left panels show the median of the vignettes. Panels and color bars are the same as in Figure 36.

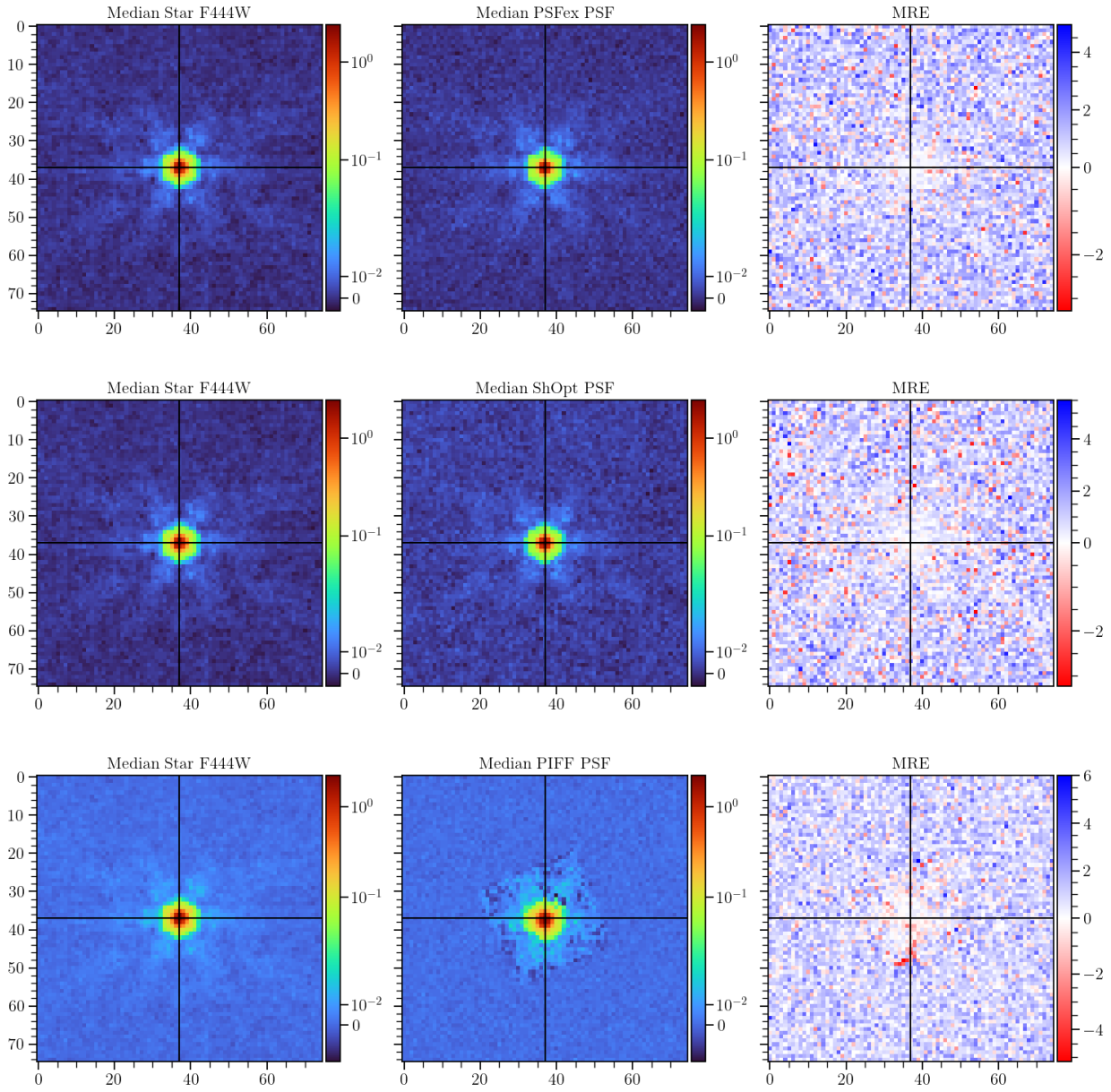



Figure 89: Evaluation of mean relative error between stars and PSF models for simulated mosaics in the F444W bandpass. Left panels show the median of the vignettes. Panels and color bars are the same as in Figure 36.

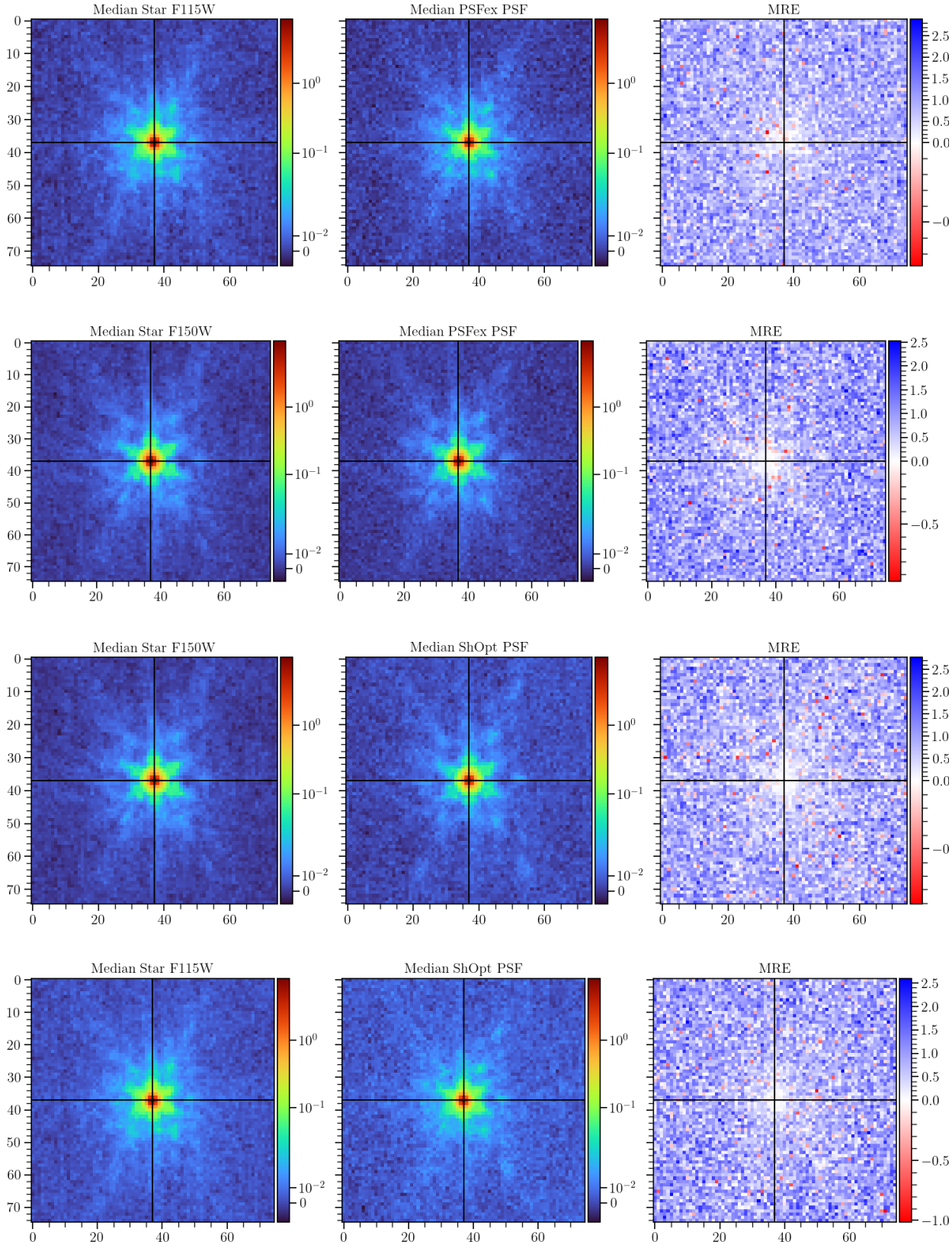



Figure 90: Mean relative error between stars and PSF models for real data mosaics in the F115W and F150W bandpasses. Color bars are the same as Figure 38.

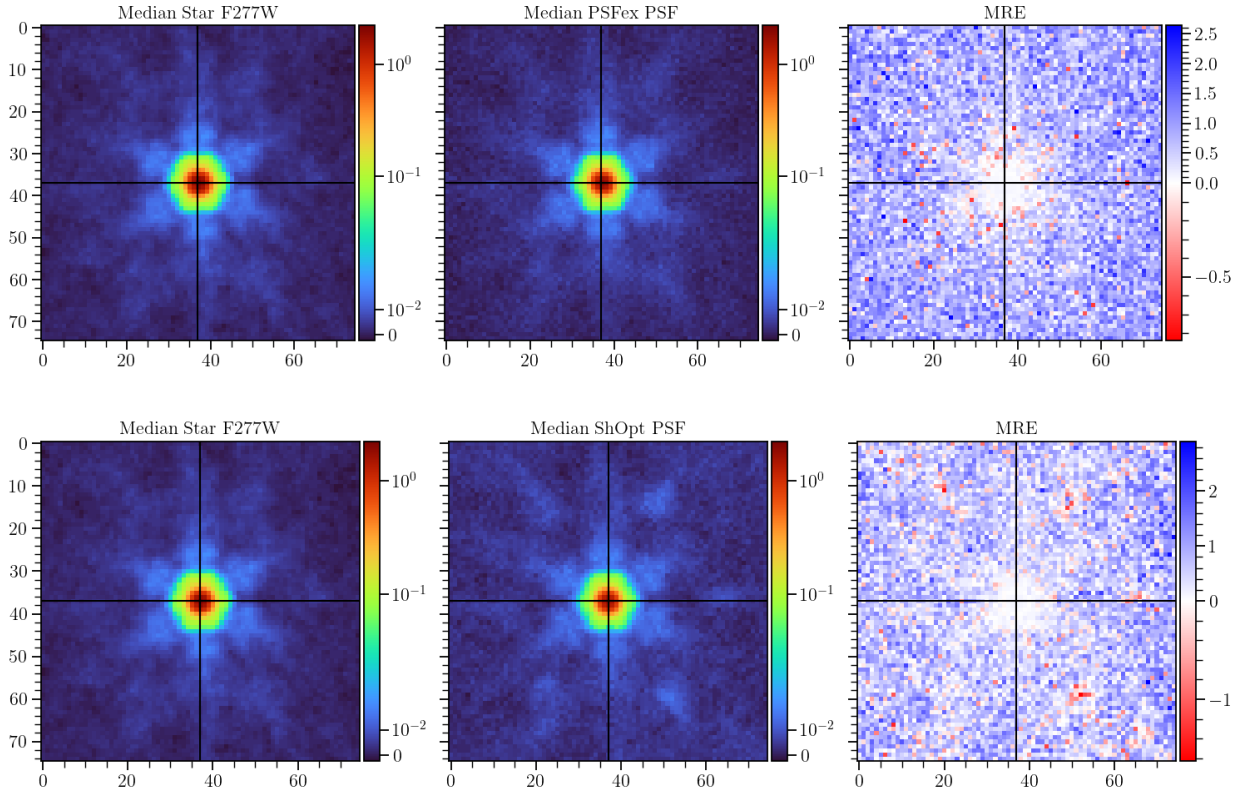



Figure 91: Evaluation of mean relative error between stars and PSF models for real data mosaics in the F277W bandpass. Panels and color bars are the same as in Figure 38.

# B Running ShOpt

```
export JULIA_NUM_THREADS=auto # On Windows set JULIA_NUM_THREADS=auto
julia shopt.jl [configdir] [outdir] [catalog.fits]
```

See our `TutorialNotebook.ipynb` or our `README.md` for more detailed setup instructions on our GitHub.

# C Testing Configs

## C.1 Source Extractor config

```
# Default configuration file for SExtractor 2.25.0
#
#-------------------------------- Catalog ------------------------------------

CATALOG_NAME     catalog.fits  # name of the output catalog
CATALOG_TYPE     FITS_LDAC
PARAMETERS_NAME  sextractor.param  # name of the file containing catalog contents

#------------------------------- Extraction ----------------------------------

DETECT_TYPE      CCD            # CCD (linear) or PHOTO (with gamma correction)
DETECT_MINAREA   8              # min. # of pixels above threshold
DETECT_MAXAREA   0              # max. # of pixels above threshold (0=unlimited)
THRESH_TYPE      RELATIVE       # threshold type: RELATIVE (in sigmas)
                                # or ABSOLUTE (in ADUs)
DETECT_THRESH    2             # <sigmas> or <threshold>,<ZP> in mag.arcsec-2
ANALYSIS_THRESH  2            # <sigmas> or <threshold>,<ZP> in mag.arcsec-2

FILTER           Y              # apply filter for detection (Y or N)?
FILTER_NAME  gauss_2.5_5x5.conv # name of the file containing the filter
FILTER_THRESH                   # Threshold[s] for retina filtering

DEBLEND_NTHRESH  32             # Number of deblending sub-thresholds
DEBLEND_MINCONT  0.005          # Minimum contrast parameter for deblending
CLEAN            Y              # Clean spurious detections? (Y or N)?
CLEAN_PARAM      1.0            # Cleaning efficiency

MASK_TYPE        CORRECT        # type of detection MASKing: can be one of
                                # NONE, BLANK or CORRECT

#------------------------------- WEIGHTing -----------------------------------

WEIGHT_TYPE    MAP_WEIGHT       # type of WEIGHTing: NONE, BACKGROUND,
                                # MAP_RMS, MAP_VAR or MAP_WEIGHT
RESCALE_WEIGHTS  Y              # Rescale input weights/variances (Y/N)?
WEIGHT_IMAGE     weight.fits    # weight-map filename
WEIGHT_GAIN      Y              # modulate gain (E/ADU) with weights? (Y/N)
WEIGHT_THRESH    0              # weight threshold[s] for bad pixels

#------------------------------- FLAGging ------------------------------------

FLAG_IMAGE       flag.fits      # filename for an input FLAG-image
FLAG_TYPE        OR             # flag pixel combination: OR, AND, MIN, MAX
                                # or MOST

#---------------------- Differential Geometry Map --------------------------

DGEO_TYPE        NONE           # Differential geometry map type: NONE or PIXEL
DGEO_IMAGE       dgeo.fits      # Filename for input differential geometry image

#------------------------------ Photometry -----------------------------------

PHOT_APERTURES   31            # MAG_APER aperture diameter(s) in pixels
PHOT_AUTOPARAMS  2, 2.5         # MAG_AUTO parameters: <Kron_fact>,<min_radius>
```

```
PHOT_PETROPARAMS 2.0, 3.5       # MAG_PETRO parameters: <Petrosian_fact>,
                                # <min_radius>
PHOT_AUTOAPERS   0.0,0.0        # <estimation>,<measurement> minimum apertures
                                # for MAG_AUTO and MAG_PETRO
PHOT_FLUXFRAC    0.5            # flux fraction[s] used for FLUX_RADIUS

SATUR_LEVEL      37000.0        # level (in ADUs) at which arises saturation
SATUR_KEY        SATURATE       # keyword for saturation level (in ADUs)

MAG_ZEROPOINT    28.086519392           # magnitude zero-point
MAG_GAMMA        4.0            # gamma of emulsion (for photographic scans)
GAIN             0.0            # detector gain in e-/ADU
GAIN_KEY         GAIN           # keyword for detector gain in e-/ADU
PIXEL_SCALE      0              # size of pixel in arcsec (0=use FITS WCS info)

#------------------------- Star/Galaxy Separation ----------------------------

SEEING_FWHM      0.07           # stellar FWHM in arcsec
STARNNW_NAME     default.nnw    # Neural-Network_Weight table filename

#------------------------------ Background -----------------------------------

BACK_TYPE        AUTO           # AUTO or MANUAL
BACK_VALUE       0.0            # Default background value in MANUAL mode
BACK_SIZE        128            # Background mesh: <size> or <width>,<height>
BACK_FILTERSIZE  3              # Background filter: <size> or <width>,<height>

BACKPHOTO_TYPE   LOCAL          # can be GLOBAL or LOCAL
BACKPHOTO_THICK  24             # thickness of the background LOCAL annulus
BACK_FILTTHRESH  0.0            # Threshold above which the background-
                                # map filter operates

#------------------------------ Check Image ----------------------------------

CHECKIMAGE_TYPE  -BACKGROUND, APERTURES     # Check-image type(s)
CHECKIMAGE_NAME  im.sub.fits, im.aper.fits  # Filename for the check-image(s)
```

## C.2 ShOpt config

```
saveYaml: true #save this file with each run

# Options: auotoencoder, PCA, smoothing.
# Make sure mode is a string with double quotes
mode: "smoothing"

# If PCA mode is enabled, how many moments do you
# want to use for your pixel grid fit
PCAterms: 50

# The size of the smoothing kernel
lanczos: 5

# For Autoencoder mode, when enabled
NNparams:
  # max number of training epochs for each pixel grid fit
  epochs: 100
  # The stopping gradient of the loss function for the pixel grid fit
  minGradientPixel: 1e-5

# For fitting analytic profile
AnalyticFitParams:
  # Stopping gradient for LBFGS on vignettes
  minGradientAnalyticModel: 1e-6
  # Stopping gradient for LBFGS on pixel grid models
  minGradientAnalyticLearned: 1e-6
  # The subset of pixels you wish to fit the analytic profile to
  analyticFitStampSize: 64
```

```
dataProcessing:
  # Filter this % of stars based off of signal to noise
  SnRPercentile: 0.33
  # Filter stars with analytic profile fit size s exceeding this value
  sUpperBound: 1
  # Filter stars with analytic profile fit size s below this value
  sLowerBound: 0.075

# What plots do you want?
plots:
  unicodePlots: true
  normalPlots:
    parametersHistogram: true
    parametersScatterplot: true
  cairomakiePlots:
    streamplots: false
  pythonPlots: false

# Degree of polynomial for spatial interpolation of PSF model
polynomialDegree: 1
# Size of pixel grid PSF model
stampSize: 130

# How many stars are you using the train versus to validate the PSF fit
training_ratio: 0.9
# Sum flux to unity true or false
sum_pixel_grid_and_inputs_to_unity: false

# stopping gradient for LFBGS used for polynomial interpolation
polynomial_interpolation_stopping_gradient: 1e-12

# Name to prefix summary.shopt
summary_name: ''
# Do you want to save storage by only storing essential information, how to reconstruct the PSF and analytic models
truncate_summary_file: true

CommentsOnRun: "** This is where you can leave comments or notes to self on the run! **"
```

## C.3 PIFF config

```
# modules and input.wcs fields, in which case, the code will use the (less
# accurate) WCS that ships with the image in the fits file.

input:

    # Input file directory
    dir: "./"

    # Input filename(s) and HDU extensions
    image_file_name: "mosaic_nircam_f277w_COSMOS-Web_i2d.fits"
    image_hdu: 1
    weight_hdu: 4

    # Input catalog and HDU extension
    cat_file_name: "mosaic_nircam_f277w_COSMOS-Web_i2d_starcat.fits"
    cat_hdu: 2

    # What columns in the catalog have things we need?
    x_col: XWIN_IMAGE
    y_col: YWIN_IMAGE
    ra_col: ALPHAWIN_J2000
    dec_col: DELTAWIN_J2000

    # The telescope pointing
    ra: 149.9303551903936
    dec: 2.380272767453749
```

```
    # Leave blank if you don't know what it is!
    # gain: 1

    # How large should the postage stamp cutouts of the stars be?
    stamp_size: 100

    # Use all cores for reading the input files
    nproc: -1

select:

    # For bright stars, weight them equivalent to snr=100 stars, not higher.
    max_snr: 100

    # Remove stars with snr < 10
    min_snr:  10

    # Reserve 15% of the stars for diagnostics
    reserve_frac: 0.15

    # Reject size outliers
    hsm_size_reject: True

psf:

    # This type of PSF will use a separate model/interp solution for each chip.
    # But all the solutions will be given in a single output file.
    type: SingleChip

    # Also use all cores when finding psf
    nproc: -1

    outliers:

        type: Chisq

        # The threshold is given in terms of nsigma equivalent
        nsigma: 4

        # Only remove at most 3% of the stars per iteration.
        max_remove: 0.03

    model:
        # This model uses a grid of pixels to model the surface brightness distribution.
        type: PixelGrid
        scale: 0.03      # NIRCam ative pixel scale
        size: 75

    interp:

        # This interpolator does some of the model solving when interpolating
        # to handle degenerate information from masking
        # and the fact that the pixels are smaller than native.
        type: BasisPolynomial
        order: 1
```

## C.4 PSFEx Config

```
# Default configuration file for PSFEx 3.17.1
# EB 2016-06-28
#

#------------------------------- PSF model ---------------------------------

BASIS_TYPE      PIXEL           # NONE, PIXEL, GAUSS-LAGUERRE or FILE
#BASIS_NUMBER   30              # Basis number or parameter
```

```
PSF_SAMPLING    0               # Sampling step in pixel units (0.0 = auto)
PSF_SIZE        261             # Image size of the PSF model
PSF_RECENTER    Y
#------------------------- Point source measurements -------------------------

CENTER_KEYS     XWIN_IMAGE,YWIN_IMAGE # Catalogue parameters for source pre-centering
PHOTFLUX_KEY    FLUX_APER(1)    # Catalogue parameter for photometric norm.
PHOTFLUXERR_KEY FLUXERR_APER(1) # Catalogue parameter for photometric error

#----------------------------- PSF variability -------------------------------

PSFVAR_KEYS     XWIN_IMAGE,YWIN_IMAGE # Catalogue or FITS (preceded by :) params
PSFVAR_GROUPS   1,1             # Group tag for each context key
PSFVAR_DEGREES  1               # Polynom degree for each group

#----------------------------- Sample selection ------------------------------

SAMPLE_AUTOSELECT  Y           # Automatically select the FWHM (Y/N) ?
SAMPLEVAR_TYPE    NONE         # File-to-file PSF variability: NONE or SEEING
SAMPLE_FWHMRANGE   1,20        # Allowed FWHM range (2.7,3.2)
SAMPLE_VARIABILITY 0.3         # Allowed FWHM variability (1.0 = 100%)
SAMPLE_MINSN       50         # Minimum S/N for a source to be used
SAMPLE_MAXELLIP    0.3         # Maximum (A-B)/(A+B) for a source to be used

#----------------------------- Output catalogs -------------------------------

OUTCAT_TYPE        FITS_LDAC      # NONE, ASCII_HEAD, ASCII, FITS_LDAC
OUTCAT_NAME        psfex_out.cat  # Output catalog filename


#------------------------------- Check-plots ---------------------------------

CHECKPLOT_DEV       PDF         # NULL, XWIN, TK, PS, PSC, XFIG, PNG,
                                # JPEG, AQT, PDF or SVG
CHECKPLOT_RES       0           # Check-plot resolution (0 = default)
CHECKPLOT_ANTIALIAS Y           # Anti-aliasing using convert (Y/N) ?
CHECKPLOT_TYPE     NONE
CHECKPLOT_NAME

#------------------------------ Check-Images ---------------------------------

CHECKIMAGE_TYPE CHI,SAMPLES,RESIDUALS,SNAPSHOTS, -SYMMETRICAL
                                # or MOFFAT,-MOFFAT,-SYMMETRICAL
CHECKIMAGE_NAME chi.fits,samp.fits,resi.fits,snap.fits, minus_symm.fits
                                # Check-image filenames
CHECKIMAGE_CUBE Y
```

## C.5 Shell Script

This is given to inform how much memory we requested from Discovery for purposes of speed testing.

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --mem=16G
#SBATCH --cpus-per-task=4
#SBATCH --time=24:00:00
#SBATCH --partition=short
#SBATCH --job-name=150_A6
#SBATCH --ntasks=1
#SBATCH --constraint=zen2           # Requesting specific CPU architecture
pwd
source /work/mccleary_group/berman.ed/minicondaInstall/bin/activate
python get_galaxy_cutouts.py -config configs/box_cutter.yaml file #file is a placeholder for a fits file
```

# D Additional ShOpt Checkplots and Outputs

## D.1 Diagnostic Material

`ShOpt` provides the following streamplots (Figure 92) to give the user an inclination toward how the PSF is changing across the field of view. We also have a Julia script, `reader.jl` that reads in the `summary.shopt` file and provides easy PSF reconstruction. If you want to do your analysis in Python, we also have Python code available for reading in `summary.shopt` files here https://github.com/EdwardBerman/sigma-Eta-Shopt-Reader.

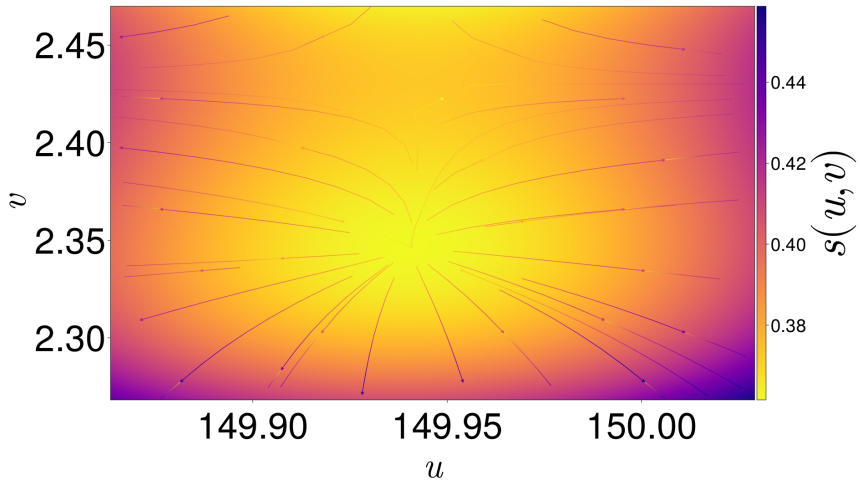
## D.2 summary.shopt

`summary.shopt` contains 6 relevant extensions. The first extension is named polynomial matrix, and it contains a 3−dimension matrix. 2−dimensions correspond to the dimensions of the input vignettes and the third dimension corresponds to the coefficients of the polynomial at that pixel. The second extension contains all data relevant to learned parameters $[s, g_1, g_2]$ as well as the $(u, v)$ coordinates at each star. We also measure the mean relative error between stars and their pixel grid fits before the polynomial interpolation step. Note that only stars that make it through all filters are contained. The third, and fourth extensions contain 3−dimensional arrays corresponding to the input vignettes and the pixel grid fits of the vignettes. The fifth extension provides flags that tell you the indices of stars that were filtered out of the final interpolation step. The sixth extension tells you how to find $[s, g_1, g_2]$ at an arbitrary $(u, v)$. There is also a mode that only outputs the first, second, and sixth extensions for reasons related to storage concern. This is enabled by default.

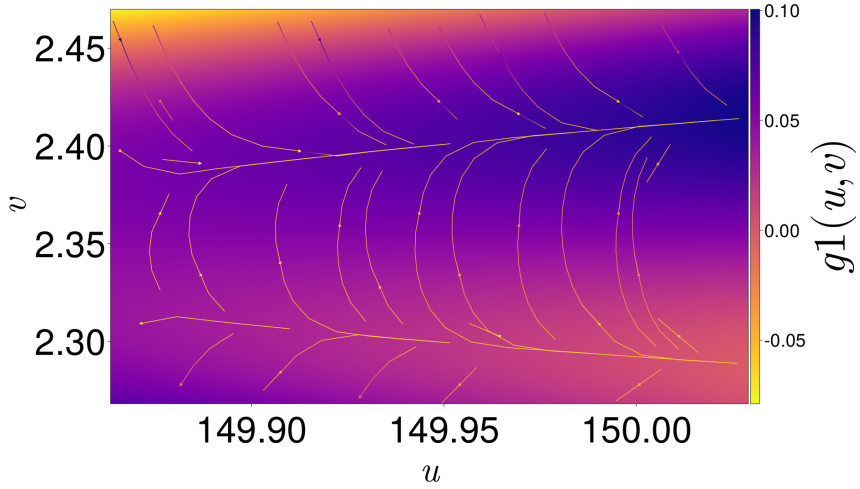
## D.3 Command Line Outputs

As an extra convenience, we give users the option to display some of the diagnostic material to the terminal using `UnicodePlots.jl`. This may be useful for less scrupulous more exploratory runs of our software or for users looking for a quick sanity check that everything ran correctly without having to navigate to an output directory and open all of the saved checkplots.

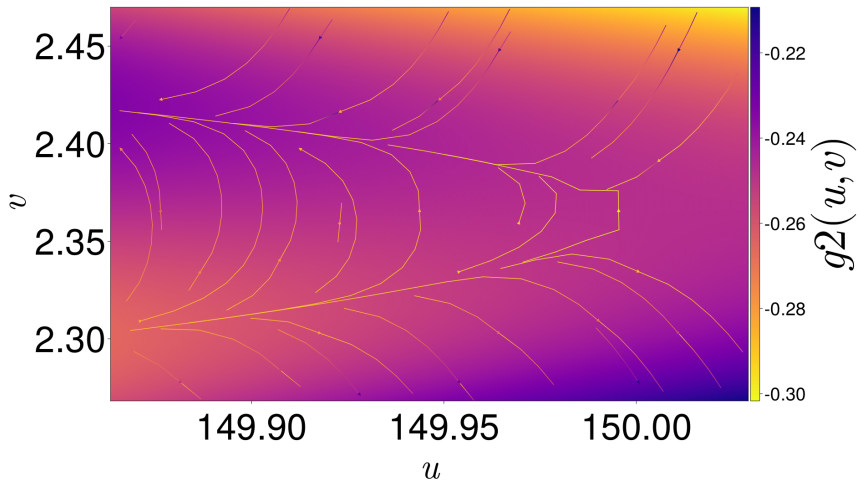
We also print out to the terminal some key information about what is happening as the program runs, including what the program is doing, how many and which stars are failing or being filtered, progress on fitting, and how long particular portions of the code took to run.




(a) $s(u, v)$




(b) $g_1(u, v)$




(c) $g_2(u, v)$

Figure 92: Stream plots demonstrating how variables $[s, g_1, g_2]$ vary across the field of view (in astrometric coordinates $(u, v)$) for the F115W simulated mosaic image. Recall that $s$ corresponds to size and $g_1, g_2$ correspond to shear.

## E Petal Diagrams

We speculate that petal diagrams may be able to approximate the spiky natures of JWST PSFS. Consider $r = A\cos(k\theta + \gamma)$, shown below in Figure 93 for different $[A, k]$ values where $\gamma = 0$. In practice, $[A, k, \gamma]$ could be learnable parameters. We could then choose some $f(r) \propto \frac{1}{r}$ such that the gray fades from black to white. We would define $f(r)$ piecewise such that it is 0 outside of the petal and decreases radially with $r$ inside the petal. The upshot of this approach is that we can just look at the learned $k$ and immediately know if our PSF captures the correct number of wings. Alternatively, Bergé et al. (2019) introduced exponential shapelets with an orthogonal separation of $r$ and $\theta$, that may also be useful.

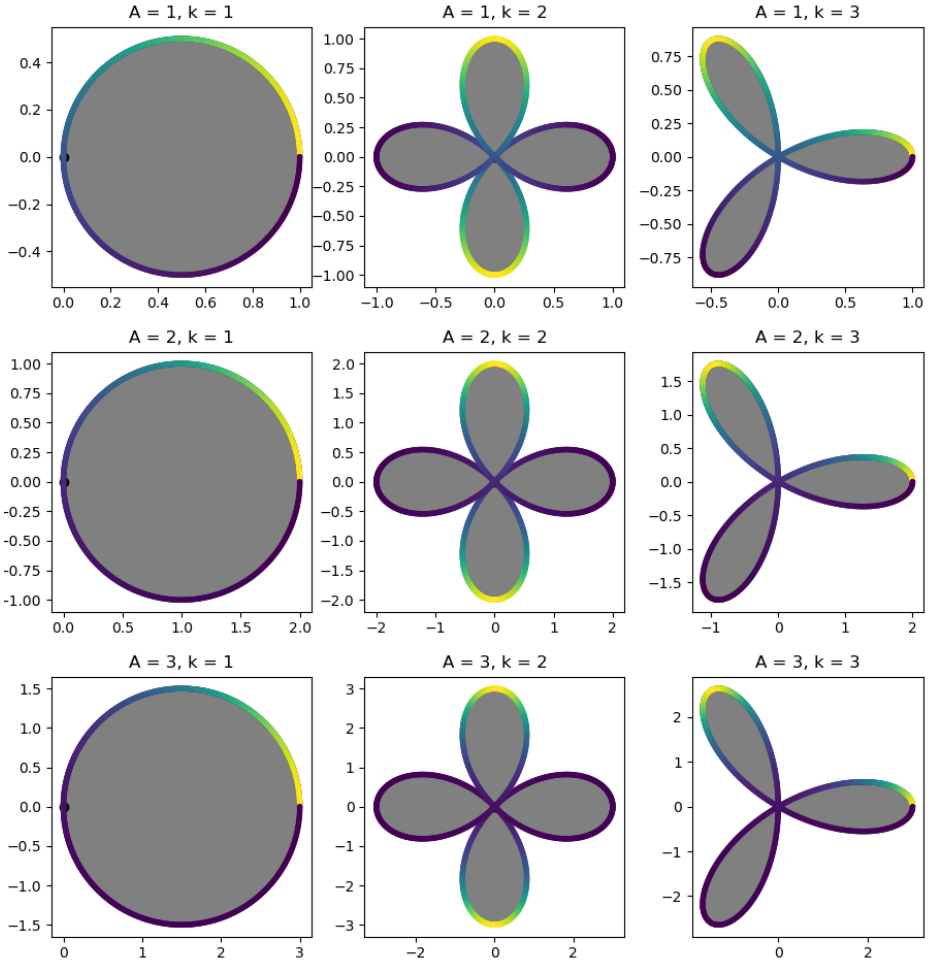



Figure 93: Petal diagrams approximating different PSFs. Different k values correspond to PSF with different numbers of spikes. The A value determines how long the spikes are. Here, all of the pixels inside the petals (gray) are set to a constant value, and everything outside is 0.