# Inferring Structural Parameters of Low-Surface-Brightness-Galaxies with Uncertainty Quantification using Bayesian Neural Networks

**Dimitrios Tanoglidis** [1] [2]  **Aleksandra Ćiprijanović** [3]  **Alex Drlica-Wagner** [3] [1] [2]

## Abstract

Measuring the structural parameters (size, total brightness, light concentration, etc.) of galaxies is a significant first step towards a quantitative description of different galaxy populations. In this work, we demonstrate that a Bayesian Neural Network (BNN) can be used for the inference, with uncertainty quantification, of such morphological parameters from simulated low-surface-brightness galaxy images. Compared to traditional profile-fitting methods, we show that the uncertainties obtained using BNNs are comparable in magnitude, well-calibrated, and the point estimates of the parameters are closer to the true values. Our method is also significantly faster, which is very important with the advent of the era of large galaxy surveys and big data in astrophysics.

## 1. Introduction

Despite their morphological diversity and complexity, the approximate light distribution of galaxies can be well-described by analytic fitting functions with a limited number of free parameters, such as their orientation, size (radius), light concentration, total brightness, etc. Measurements of these parameters allow for a quantitative comparison of different galaxy populations and the derivation of empirical scaling relations (e.g., Courteau et al., 2007), which in turns facilitates the testing of galaxy formation models.

Traditionally, these parameters are measured using galaxy profile fitting software (two widely used options being `GALFIT`; Peng et al. (2002) and `Imfit`; Erwin (2015)) that performs a $\chi^2$ minimization between the chosen analytic model and a given galaxy image, to derive the best-fit

parameters. Despite their success, these codes have their limitations, too: they are not optimized to fit a large number of galaxies quickly, and they usually require some manual intervention (for example, the selection of good initial model parameters). The low speed is an even more significant problem if one wants to obtain accurate estimates of the uncertainties associated with those measurements, for example via bootstrap resampling, or by using a Markov-Chain Monte Carlo (MCMC) approach to sample the parameter posterior distribution.

Large galaxy surveys, such as the Dark Energy Survey (DES)[1] and the upcoming Legacy Survey of Space and Time (LSST)[2] on the Vera C. Rubin Observatory, observe hundreds of millions (the former) to tens of billions (the latter) of galaxies. With the advent of these surveys, fast, automated, and reliable methods for measuring the structural parameters of galaxies are needed. Deep learning methods are well-suited to tackle this problem since, once trained, they are able to make predictions on new, unseen, examples very quickly.

Indeed, several works (e.g., Tuccillo et al., 2018; Aragon-Calvo & Carvajal, 2020; Li et al., 2022) have demonstrated that Convolutional Neural Networks (CNNs), trained on simulated galaxy images are significantly faster, and almost as accurate as the standard profile fitting methods in predicting galaxy parameters. However, those works used standard, deterministic, neural networks, that output single-point estimates and thus are unable to quantify the uncertainty associated with their predictions.

A rigorous uncertainty quantification is imperative for studies of challenging, low signal-to-noise objects, such as low-surface-brightness galaxies (LSBGs). LSBGs, defined as galaxies with a central brightness at least a magnitude fainter than that of the ambient dark sky, are challenging to observe and galaxy surveys have only recently started to produce large LSBG catalogs (Greco et al., 2018; Tanoglidis et al., 2021; Zaritsky et al., 2022), although they are expected to dominate the galaxy population. LSBGs are a target of future surveys, in the quest of understanding the galaxy

[1]Department of Astronomy & Astrophysics, University of Chicago, Chicago IL 60637, USA [2]Kavli Institute for Cosmological Physics, Chicago IL 60637, USA [3]Fermi National Accelerator Laboratory, Batavia, IL 60510, USA. Correspondence to: Dimitrios Tanoglidis <dtanoglidis@uchicago.edu>.

---

[1]https://www.darkenergysurvey.org/
[2]https://https://www.lsst.org/

formation process in a relatively unexplored regime.

In this work, we explore the use of Bayesien Neural Networks (BNNs; e.g., Valentin Jospin et al., 2020) for the problem of LSBG structural parameter estimation with simultaneous uncertainty quantification. BNNs output posterior probability distributions instead of point estimates for their predictions, and thus they naturally offer a way to quantify the uncertainties associated with neural network predictions. Specifically, we use a simulated dataset of DES-like LSBGs, to train, validate, and test a convolutional BNN model and compare the speed and accuracy of its predictions with those obtained using $\mathtt{pyImfit}$[3], for a single-component Sérsic light-profile model.

Our code and simulated data related to this work are available at: https://github.com/dtanoglidis/BNN_LSBGs_ICML.
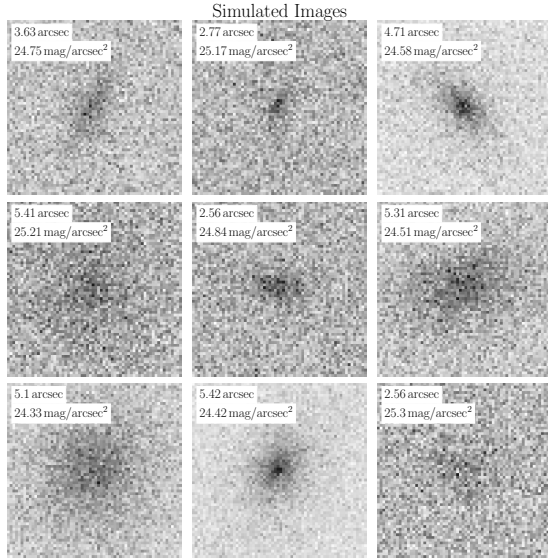
## 2. Simulated Data



Simulated Images

*Figure 1.* Examples of simulated LSBG images. The inset text refers to the effective radius, $r_e$, (top) and the surface brightness, $I_e$ (bottom).

We use $\mathtt{PyImfit}$ to create a simulated dataset of 170,000 LSBG images. Each image has dimensions $64 \times 64$ pixels, and it has two components: a uniform background $I(x, y) = I_{\mathrm{sky}}$, and a Sérsic function (Sérsic, 1963) that describes the surface-brightness profile of the galaxies:

$$I_{\mathrm{gal}}(r) = I_e \exp \left\{ -b_n \left[ \left( \frac{r}{r_e} \right)^{1/n} - 1 \right] \right\}, \quad (1)$$

---

[3]$\mathtt{pyImfit}$ is a Python wrapper around $\mathtt{Imfit}$.

where $r = (x^2 + y^2/q^2)^{1/2}$, $(x, y)$ are the coordinates with origin at the center of the image, and $q$ is the axis ratio. The axis ratio is connected to the ellipticity as $q = 1 - \epsilon$. The other free parameters of the model are: the effective half-light radius, $r_e$, the surface brightness at the effective radius, $I_e$, the Sérsic index, $n$, that controls the shape of the light distribution, and the position angle that defines the orientation of the galaxy profile. As for the value of $b_n$ (not a free parameter of the model), $\mathtt{pyImfit}$ uses the approximation by Ciotti & Bertin (1999).

We want our simulated images to resemble real LSBG images. For that reason we sample parameters uniformly, from a range that roughly corresponds to the bulk of the LSBGs discovered by DES, as described in Tanoglidis et al. (2021):

- Position angle, PA $\in [0, 180]$ degrees,
- Ellipticity, $\epsilon \in [0.05, 0.7]$,
- Sérsic index, $n \in [0.5, 1.5]$,
- Surface brightness, $I_e \in [24.3, 25.5]$ mag/arcsec$^2$,
- Effective radius, $r_e \in [2.5, 6.0]$ arcsec.

Furthermore, we assume a pixel to angular scale conversion 1 pix = 0.263 arcsec (as in DES), and background sky surface brightness $I_{\mathrm{sky}} = 22.23$ mag/arcsec$^2$ (Neilsen et al., 2016). At each pixel we randomly assign a number of photons (counts) drawn from a Poisson distribution with mean the one predicted from the total surface brightness model $I_{\mathrm{tot}} = I_{\mathrm{gal}} + I_{\mathrm{sky}}$. In Fig. 1 we present a small subset of the simulated galaxy images.

## 3. Model

### 3.1. Bayesian Neural Networks (BNNs)

Standard neural networks, once trained, output a single point estimate prediction every time the same example is presented to the network. Thus, *deterministic* neural networks are unable to capture uncertainties in their predictions. In BNNs, the single weights are being replaced by appropriate probability distributions, that can be subsequently used to provide a measure of how (un)certain a model is in its predictions.

Given a training dataset, $\mathcal{D} = (\mathbf{x}, \mathbf{y})$, training a BNN consists of finding the posterior distribution of the weights, $p(w|\mathcal{D})$. The exact evaluation of this posterior distribution is computationally intractable, since it requires multidimensional integration over the weight values. One approach (variational inference; VI) is to approximate the true posterior with a distribution $q(w|\theta)$ of a known form (e.g., a Gaussian), with free parameters $\theta$ to be learned. The goal is to select the parameters to minimize the difference between the true and approximate posteriors. It can be shown that this is equivalent to minimizing the (negative) evidence

lower bound (ELBO) loss function (e.g., Blundell et al., 2015):

$$\mathcal{L}(\mathcal{D}, \theta) = \mathbb{E}_{q(w|\theta)}\left[\log q(w|\theta) - \log p(w)p(\mathcal{D}|w)\right]. \quad (2)$$

Once trained, random weights can be drawn from the approximate posterior $q(w|\theta)$, and the network can give predictions on new examples presented to it.

### 3.2. Implementation and Training

Our BNN architecture consists of five (probabilistic) 2D convolutional layers, each one followed by a Max Pooling layer, and a (probabilistic) dense layer following the convolutional part. The model output is a multivariate ($n = 5$ dimensions, equal to the five free parameters of the Sérsic model described in Sec. 2) Gaussian, with full covariance. For the interested reader, we present a schematic overview and more details about the architecture in Appendix A.

We implement our network using the `Keras` and `Tensorflow Probability` (TFP) Python libraries. Before training, we randomly split the full simulated dataset to a training (150k), a validation (10k), and a test (10k) set. We perform training with a learning rate $\eta = 0.2$ (`Adadelta` optimizer), for 150 epochs, using a batch size of 64. During training we observed no signs of overfitting. We utilized the 25 GB high-RAM Nvidia P100 GPUs available through the Google Colab Pro. The training took approximately three hours to complete.

## 4. Results

### 4.1. Parameter posteriors

In Fig. 2 we present the predicted posterior distributions for the five parameters of the Sérsic model, described in Sec. 2, for a simulated galaxy at the bright end of the surface brightness distribution ($I_e = 24.4$ mag/arcsec$^2$, panel (a)), and one at the faint end ($I_e = 25.3$ mag/arcsec$^2$, panel (b)). We present the predictions of the BNN model (red contours) and those from the `pyImfit` model, using two different estimation methods: bootstrap resampling (green contours), and Markov chain Monte Carlo (MCMC; blue contours).

To get the BNN posteriors, we stack together the output distributions from 400 forward passes (predictions) of the model, for each one of the LSBG images. We see that the constraints on the parameters are tighter (as in the case of the brighter LSBG) or comparable (as in the case of the fainter LSBG) to those obtained using `pyImfit`. Although we present only two examples here, we have confirmed that this is true for a larger number of randomly selected LSBG images.

In terms of speed, obtaining the posterior distribution for each one of the examples using BNNs was significantly
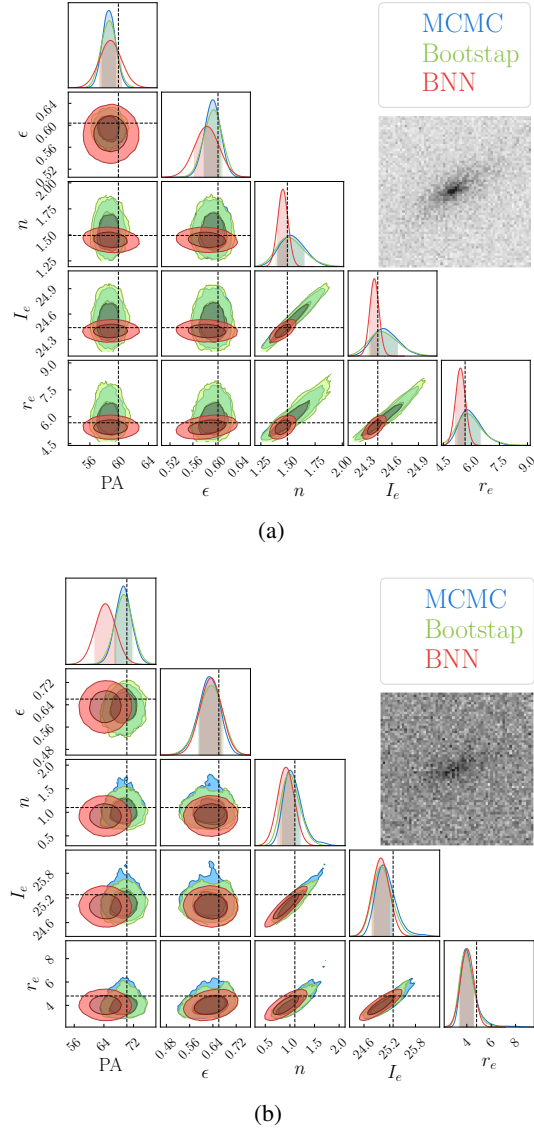


(a)



(b)

*Figure 2.* Predicted posterior distributions of structural parameters for an LSBG in the bright end ($I_e = 24.4$ mag/arcsec$^2$, panel (a)), and one in the faint end ($I_e = 25.3$ mag/arcsec$^2$, panel (b)) of the surface brightness range we consider in this work. The dashed lines indicate the true parameter values. The plots were created using `ChainConsumer` (Hinton, 2016).

faster than running MCMC ($\sim 1$ minute vs $\sim 6$ minutes), and comparable in time to the bootstrap method. The real gain in time comes when one wants to process a large number of galaxy images simultaneously; for example, obtaining full posterior estimates for 1000 LSBGs using BNNs takes $\sim 7$ minutes on our machine, while processing the same number of images using `pyImfit` and bootstrap resampling would had taken $\sim 16$ hours.
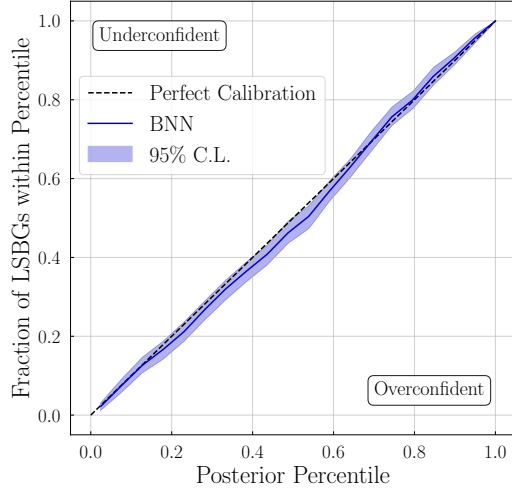
*Figure 3.* Calibration curve of the BNN posterior for the effective radius parameter, $r_e$. Within the statistical uncertainty, the BNN curve agrees with the diagonal line that indicates perfect calibration.



*Figure 4.* Comparison of the effective radius parameter, $r_e$, predictions from the BNN model (blue dots) and from `pyImfit` (red dots) versus the true values.

## 4.2. Calibration

We have showed that BNNs fit Sérsic model parameters with tighter or similar uncertainties to those produced by `pyImfit`. In order to be interpreted as confidence intervals, we have to demonstrate that a $x\%$ interval contains the true value $x\%$ of the time – in other words, that the posterior is well-calibrated.

To investigate that, we consider the parameter posterior predictions on 1000 simulated LSBGs drawn from the test set. Following Wagner-Carena et al. (2021); Park et al. (2021), we calculate, at different posterior percentile levels, the fraction of LSBGs with true values within the limits of that percentile interval. For well-calibrated posteriors, those two quantities (percentile and fraction) should be equal. Here, we show only the (marginalized) posterior for the effective radius, $r_e$. However, similar results are found for the other parameters; the interested reader can see those plots in Appendix B. As we can see in Fig. 3, the posterior produced by the BNN is statistically consistent with being perfectly calibrated (the error band was calculated by performing a bootstrap resampling of the test set used for the calculations).

## 4.3. Comparison of point estimates

We have demonstrated that the BNN model outputs well-calibrated errors, and we have seen examples where we compared the output parameter posteriors from the BNN and the `pyImfit` algorithm. We now compare the point estimate (mean) prediction from the BNN with the best fit
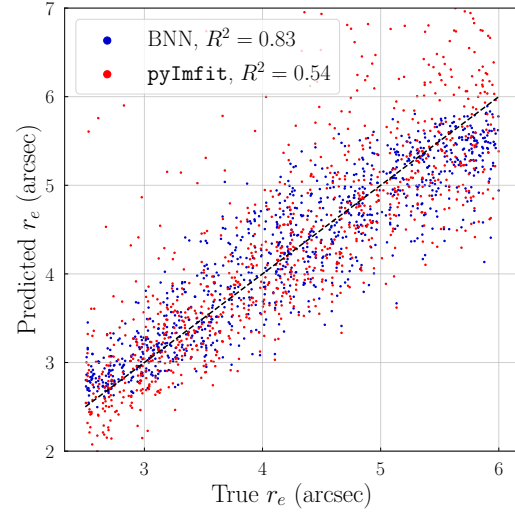
parameter output from `pyImfit`.

In Fig. 4 we plot the true value of the effective radius parameter (for the same 1000 simulated LSBGs as in the previous section) vs the predicted one, using both methods. The point estimates produced by the BNN method tend to be closer to the true value, as indicated by the higher coefficient of determination ($R^2 = 0.83$ for BNN vs $R^2 = 0.54$ for `pyImfit`), and it performs significantly better for higher effective radius values. In Appendix B we present similar plots for the other Sérsic model parameters.

## 5. Discussion and Future Work

We have used a Bayesian Neural network model to predict structural parameters of LSBGs in simulated galaxy images. We compare the posterior parameter predictions from the BNN method with those from a profile-fitting algorithm (`pyImfit`) for simulated LSBGs and we show that the BNN gives comparable or even tighter parameter constraints.

We furthermore show that the uncertainties estimated using the BNN method are well calibrated, and that, for a sample of simulated LSBGs, the BNN gives better point-estimate parameter predictions (higher coefficient of determinations) compared to those from `pyImfit`.

A significant strength of our BNN method is its speed. For example, it can predict the full posterior distribution of the five-parameter Sérsic model for 1000 LSBGs images within $\sim 7$ minutes (on the machine used here); using `pyImfit` and the bootstrap resampling methods to get parameter con-

straints for the same number of images, would require $\sim 16$ hours.

An important next step, which we plan to address in future work, is to test the performance of our method on real LSBG images and investigate ways to improve it if necessary (for example by re-training on real data, as in Tuccillo et al. (2018)). Indeed, the case presented here, where both training and testing was done on very simple simulated data, can be significantly different from real data. However, some preliminary investigation on applying the model trained here on real LSBGs has shown promising results.

Other areas of future investigation include testing different BNN architectures, testing the performance of the model on data outside of the training range, and a more rigorous comparison of the performance (parameter constraints and speed) between BNNs and `pyImfit`.

## References

Aragon-Calvo, M. A. and Carvajal, J. C. Self-supervised learning with physics-aware neural networks - I. Galaxy model fitting. *Monthly Notices of the Royal Astronomical Society*, 498(3):3713–3719, November 2020. doi: 10. 1093/mnras/staa2228.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight Uncertainty in Neural Networks. *arXiv e-prints*, art. arXiv:1505.05424, May 2015.

Ciotti, L. and Bertin, G. Analytical properties of the $R^{1/m}$ law. *Astronomy & Astrophysics*, 352:447–451, December 1999.

Courteau, S., Dutton, A. A., van den Bosch, F. C., MacArthur, L. A., Dekel, A., McIntosh, D. H., and Dale, D. A. Scaling Relations of Spiral Galaxies. *The Astrophysical Journal*, 671(1):203–225, December 2007. doi: 10.1086/522193.

Erwin, P. IMFIT: A Fast, Flexible New Program for Astronomical Image Fitting. *The Astrophysical Journal*, 799 (2):226, February 2015. doi: 10.1088/0004-637X/799/2/ 226.

Greco, J. P., Greene, J. E., Strauss, M. A., Macarthur, L. A., Flowers, X., Goulding, A. D., Huang, S., Kim, J. H., Komiyama, Y., Leauthaud, A., Leisman, L., Lupton, R. H., Sifón, C., and Wang, S.-Y. Illuminating Low Surface Brightness Galaxies with the Hyper Suprime-Cam Survey. *The Astrophysical Journal*, 857(2):104, April 2018. doi: 10.3847/1538-4357/aab842.

Hinton, S. R. ChainConsumer. *The Journal of Open Source Software*, 1:00045, August 2016. doi: 10.21105/joss. 00045.

Li, R., Napolitano, N. R., Roy, N., Tortora, C., La Barbera, F., Sonnenfeld, A., Qiu, C., and Liu, S. Galaxy Light Profile Convolutional Neural Networks (GaLNets). I. Fast and Accurate Structural Parameters for Billion-galaxy Samples. *The Astrophysical Journal*, 929(2):152, April 2022. doi: 10.3847/1538-4357/ac5ea0.

Neilsen, Eric H., J., Bernstein, G., Gruendl, R., and Kent, S. Limiting Magnitude, $\tau$, $t_{eff}$, and Image Quality in DESY Year 1. 3 2016. doi: 10.2172/1250877.

Park, J. W., Villar, A., Li, Y., Jiang, Y.-F., Ho, S., Yao-Yu Lin, J., Marshall, P. J., and Roodman, A. Inferring Black Hole Properties from Astronomical Multivariate Time Series with Bayesian Attentive Neural Processes. *arXiv e-prints*, art. arXiv:2106.01450, June 2021.

Peng, C. Y., Ho, L. C., Impey, C. D., and Rix, H.-W. Detailed Structural Decomposition of Galaxy Images. *The Astronomical Journal*, 124(1):266–293, July 2002. doi: 10.1086/340952.

Sérsic, J. L. Influence of the atmospheric and instrumental dispersion on the brightness distribution in a galaxy. *Boletin de la Asociacion Argentina de Astronomia La Plata Argentina*, 6:41–43, February 1963.

Tanoglidis, D., Drlica-Wagner, A., Wei, K., et. al, and (DES Collaboration). Shadows in the Dark: Low-surface-brightness Galaxies Discovered in the Dark Energy Survey. *The Astrophysical Journal Supplement series*, 252 (2):18, February 2021. doi: 10.3847/1538-4365/abca89.

Tuccillo, D., Huertas-Company, M., Decencière, E., Velasco-Forero, S., Domínguez Sánchez, H., and Dimauro, P. Deep learning for galaxy surface brightness profile fitting. *Monthly Notices of the Royal Astronomical Society*, 475(1):894–909, March 2018. doi: 10.1093/mnras/stx3186.

Valentin Jospin, L., Buntine, W., Boussaid, F., Laga, H., and Bennamoun, M. Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users. *arXiv e-prints*, art. arXiv:2007.06823, July 2020.

Wagner-Carena, S., Park, J. W., Birrer, S., Marshall, P. J., Roodman, A., Wechsler, R. H., and LSST Dark Energy Science Collaboration. Hierarchical Inference with Bayesian Neural Networks: An Application to Strong Gravitational Lensing. *The Astrophysical Journal*, 909 (2):187, March 2021. doi: 10.3847/1538-4357/abdf59.

Zaritsky, D., Donnerstein, R., Karunakaran, A., Barbosa, C. E., Dey, A., Kadowaki, J., Spekkens, K., and Zhang, H. Systematically Measuring Ultra-Diffuse Galaxies (SMUDGes). III. The Southern SMUDGes Catalog. *arXiv e-prints*, art. arXiv:2205.02193, May 2022.

## A. BNN Architecture

In Fig. 5 we present a schematic overview of the BNN architecture used in this work. As we mentioned in the main text, we used the `Keras`[4] library on a `TensorFlow`[5] backend, and the `Tensorflow Probability`[6] extension of it, for the probabilistic layers.

The architecture consists of five probabilistic convolutional layers (`convolution2DFlipout`, provided by `TensorFlow Probability`). The number of filters and the kernel size used in each layer can be seen in the figure. Each convolutional layer is followed by a Max Pooling layer. After flattening we have a dense layer (`DelseFlipout`). The output is a multidimensional Gaussian (the five parameters our model tries to learn), with full covariance that allows to capture the correlations between the parameters.

---

[4]https://keras.io/
[5]https://www.tensorflow.org/
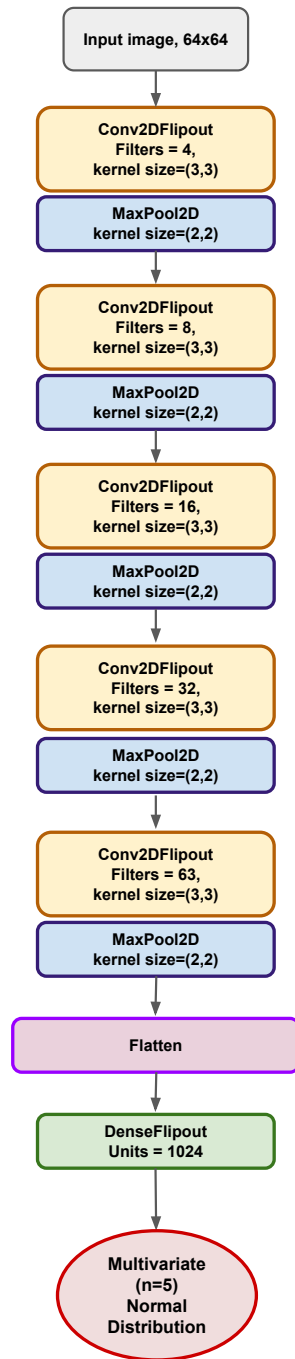[6]https://www.tensorflow.org/probability

*Figure 5.* Schematic representation of the BNN architecture used in this work.

# B. Calibration and Point Estimate Comparison Plots

In Sec. 4.2, we presented the calibration plot for the BNN posterior for the effective radius, $r_e$, and the comparison of the BNN point-estimate predictions with those coming from pyImfit, for the same parameter. We showed that the BNN errors are well-calibrated, and that the BNN point-estimates are closer to the true value compared to those of the pyImfit for the effective radius parameter.

The figures presented in this Appendix demonstrate that these conclusions hold for other model parameters, too, with the notable exception of the position angle (PA). For the PA, the network seems to be confused by the artificial discontinuity at the 0 to 180 degrees boundary (see Fig. 7d). In future iterations of this work, we plan to remove this discontinuity by reparameterizing the position angle.
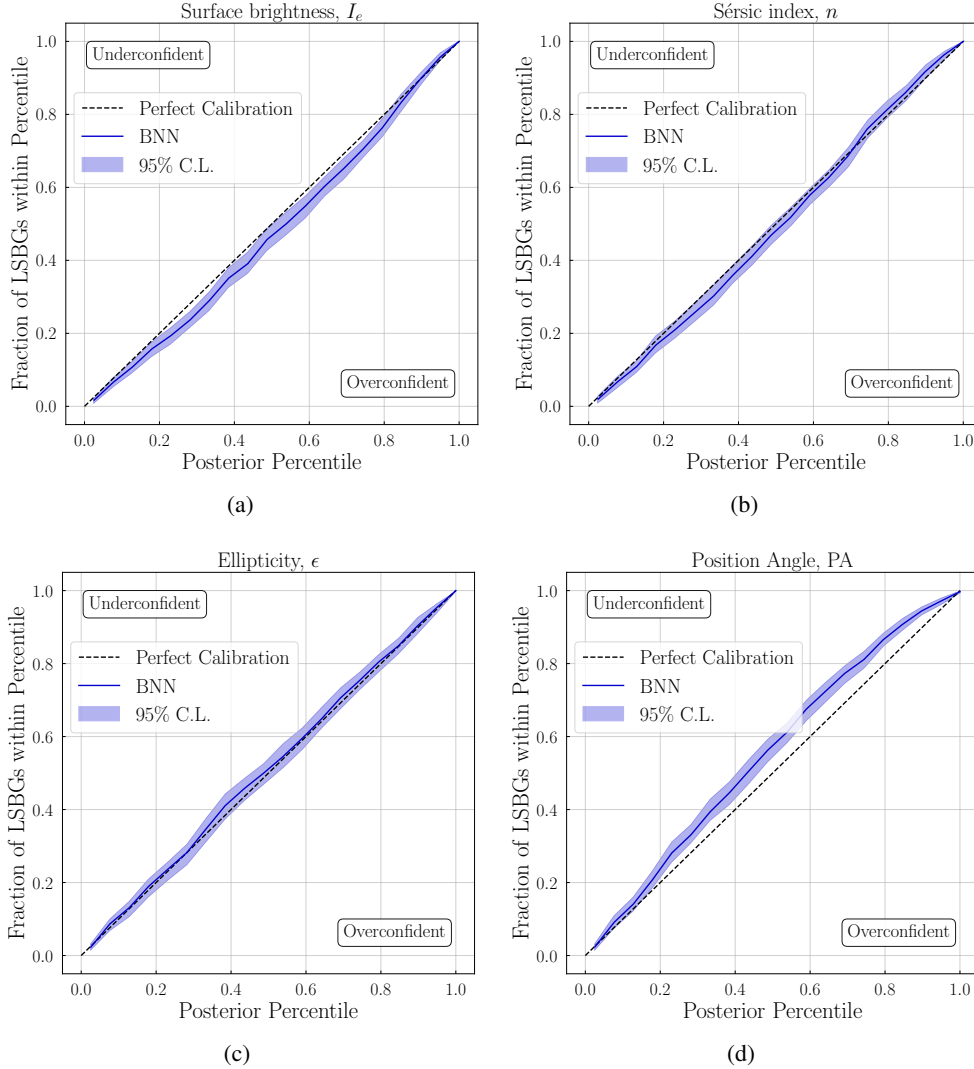


*Figure 6.* Calibration curves for the BNN posteriors of the four parameters of the Sérsic model, not presented in the main text. Expect for the position angle (PA) parameter, which the BNN seems to give slightly underconfident predictions, the calibration curves of the other parameters indicate perfect calibration, within the statistical uncertainty.
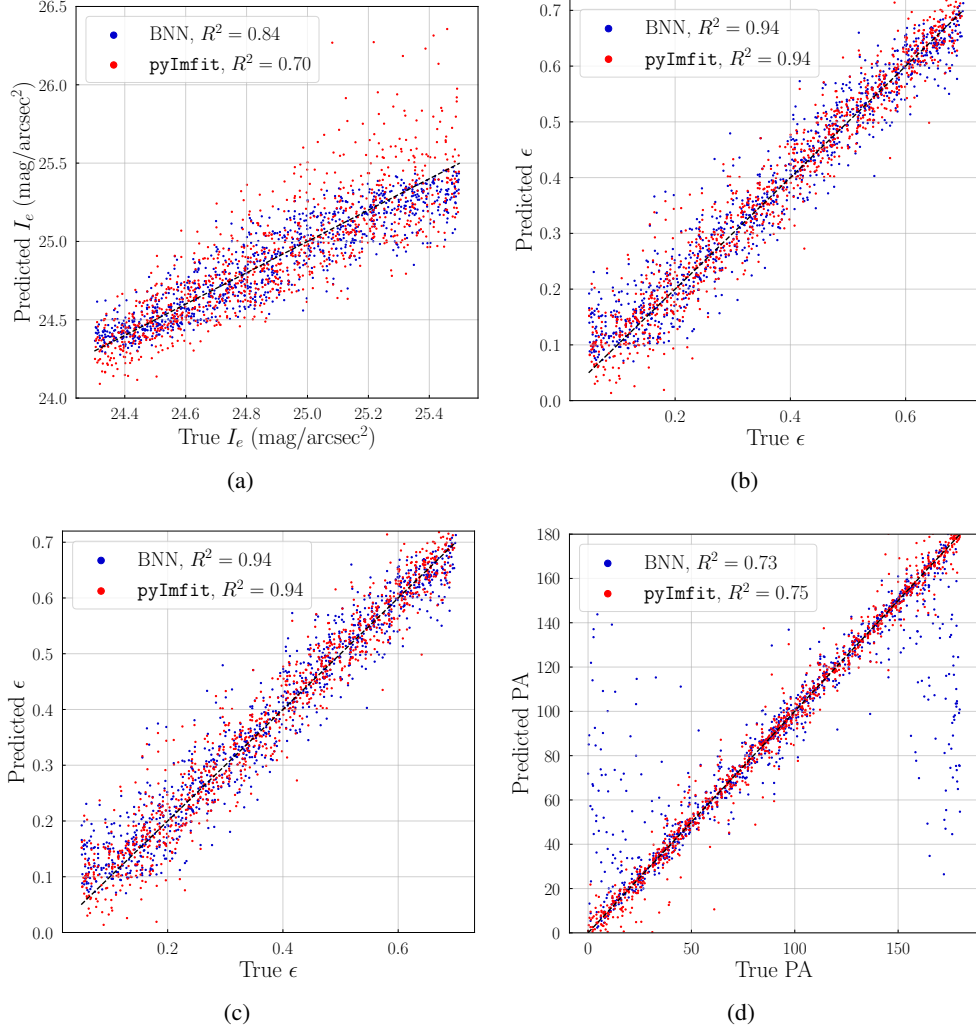
(a)

(b)

(c)

(d)

*Figure 7.* Comparison of the parameter predictions (point estimates) from the BNN model (blue dots) and pyImfit (red dots) versus the ground truth values, for the Sérsic model parameters not presented in the main text.