




PNet—A Deep Learning Based Photometry and Astrometry Bayesian Framework

Rui Sun¹, Peng Jia^{1,2,3} , Yongyang Sun¹, Zhimin Yang¹, Qiang Liu¹, and Hongyan Wei¹

¹ College of Electronic Information and Optical Engineering, Taiyuan University of Technology, Taiyuan, 030024, People's Republic of China
robinmartin20@gmail.com

² Peng Cheng Lab, Shenzhen, 518066, People's Republic of China

³ Department of Physics, Durham University, DH1 3LE, UK

Received 2023 June 24; revised 2023 October 5; accepted 2023 October 9; published 2023 November 9

Abstract

Time-domain astronomy has emerged as a vibrant research field in recent years, focusing on celestial objects that exhibit variable magnitudes or positions. Given the urgency of conducting follow-up observations for such objects, the development of an algorithm capable of detecting them and determining their magnitudes and positions has become imperative. Leveraging the advancements in deep neural networks, we present PNet, an end-to-end framework designed not only to detect celestial objects and extract their magnitudes and positions, but also to estimate the photometric uncertainty. PNet comprises two essential steps. First, it detects stars and retrieves their positions, magnitudes, and calibrated magnitudes. Subsequently, in the second phase, PNet estimates the uncertainty associated with the photometry results, serving as a valuable reference for the light-curve classification algorithm. Our algorithm has been tested using both simulated and real observation data, demonstrating the ability of PNet to deliver consistent and reliable outcomes. Integration of PNet into data-processing pipelines for time-domain astronomy holds significant potential for enhancing response speed and improving the detection capabilities for celestial objects with variable positions and magnitudes.

Unified Astronomy Thesaurus concepts: [Time domain astronomy \(2109\)](#); [Photographic astrometry \(1227\)](#); [Bayesian statistics \(1900\)](#); [CCD photometry \(208\)](#); [Neural networks \(1933\)](#)

1. Introduction

In recent years, time-domain astronomy has emerged as an active research field. With the availability of telescopes possessing a wide field of view and high image quality, it has become feasible to capture images of celestial objects at regular intervals, yielding a substantial amount of observational data on a daily basis. Among this vast data set are numerous celestial objects that necessitate frequent or immediate follow-up observations, such as tidal disruption events, near-Earth objects, superflares, and microlensing events. Consequently, there is a pressing need to develop an algorithm capable of swiftly detecting these events. Since these events primarily involve changes in the positions and magnitudes of celestial objects, the algorithm must possess the capability to detect celestial objects and conduct precise photometry and astrometry measurements. Furthermore, given that images of celestial objects are susceptible to various sources of noise, the algorithm should also be able to estimate the uncertainties associated with the photometry results, enabling further in-depth analysis.

Numerous pipelines have been proposed to meet these requirements, typically comprising the following key steps:

1. Target detection: The positions of potential celestial object candidates are determined.
2. Target classification: True celestial objects are identified from the pool of candidates, and are further categorized into different types.
3. Target information extraction: Magnitudes, positions, and distributions of the celestial objects are obtained.

Previous studies have introduced a variety of algorithms to establish the conventional data-processing pipeline. Typically, target-detection algorithms such as SExtractor or simplexy have been used to identify potential celestial objects from the original observational images (Bertin & Arnouts 1996; Lang et al. 2010). Subsequently, these identified targets undergo classification algorithms that aim to distinguish true celestial objects from the candidate pool (Cabrera-Vives et al. 2017; Duev et al. 2019; Jia et al. 2019; Agarwal et al. 2020; Turpin et al. 2020). The resulting information regarding these celestial objects is then processed through photometry, astrometry, morphology classification, or segmentation algorithms (Khramtsov et al. 2019; Boucaud et al. 2020; Hausen & Robertson 2020; Domínguez Sánchez et al. 2022; Casetti-Dinescu et al. 2023). However, the classical data-processing pipeline follows a sequential structure, wherein all the processes are executed in sequence. Consequently, the overall performance of the data-processing pipeline is limited by the performance of each individual algorithm used. For example, if celestial objects are not detected by the source-detection algorithm, it becomes impossible to extract information related to those targets. It should be noted that contemporary source-detection algorithms possess numerous adjustable parameters, requiring the expertise of experienced scientists to properly set them. In addition, because the detection results are sensitive to environmental conditions, frequent human intervention is required to obtain effective results. Therefore, it is necessary to develop an end-to-end framework that is not only able to detect celestial objects, but also to extract their information automatically and robustly.

Algorithms based on deep neural networks (DNN) for celestial object detection have attracted considerable attention in recent years. One key advantage is that these algorithms enable end-to-end learning, allowing the neural network to



Original content from this work may be used under the terms of the [Creative Commons Attribution 4.0 licence](#). Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

directly acquire the ability to detect celestial objects. Different tasks can be addressed by designing and deploying DNNs with specific architectures (Ren et al. 2015; Liu et al. 2021b; Ge et al. 2021). In this study, our focus is on detecting point-like celestial objects in sparse star fields and extracting their positions and magnitudes, as this is a crucial prerequisite for studying these objects in time-domain astronomy. It is worth noting that extended targets and dense star fields (the distance between stars are less than twice the full width at half magnitude of the point-spread function, PSF) may benefit from multicolor images and other relevant neural networks for better detection and classification (González et al. 2018; Farias et al. 2020; Cheng et al. 2021; Jia et al. 2022; Yu et al. 2022; Jia et al. 2023b; Andrew 2023), or from methods specifically designed for dense star fields (Liu et al. 2021a; Hansen et al. 2022). Moreover, transients, such as supernovae in galaxies, can be further processed using methods based on the image difference or techniques developed based on temporal sequences of images (Kessler et al. 2015; Wright et al. 2015; Zackay et al. 2016; Sánchez et al. 2019; Gómez et al. 2020; Mong et al. 2020; Hu et al. 2022; Makhlof et al. 2022). A previous study by Jia et al. (2020) has introduced a faster-RCNN-based framework for detecting point-like celestial objects, which was successfully applied to images captured by wide-field optical telescopes and the Lobster-Eye telescope (Jia et al. 2023a). However, in real applications, three key challenges need to be addressed:

1. The current framework does not provide apparent magnitudes as part of its output. Apparent magnitudes play a crucial role in various tasks, such as exoplanet observations or studying superflares from stars. Therefore, it is essential to integrate a photometry algorithm into the detection framework to accurately estimate the apparent magnitudes of the celestial targets.
2. Contemporary DNN-based target-detection algorithms define the positions of celestial objects using bounding boxes, which are rectangular boxes that approximate the shape of the objects. However, to match these celestial objects with catalogs and perform further analysis, we need to determine the precise centers of the celestial object images. Hence, it is necessary to incorporate an astrometry method into the detection framework to accurately estimate the centers of different celestial objects.
3. The current framework lacks the ability to estimate uncertainties associated with magnitude estimation. Because most neural networks provide point estimates for a given input, they directly output regression values without accounting for uncertainties. However, uncertainties are vital for subsequent tasks, such as light-curve classification. Therefore, it is crucial to develop a method that can estimate the uncertainties introduced by magnitude estimations.

To enhance the suitability of our faster-RCNN-based astronomical detection algorithm for integration into time-domain astronomy data-processing pipelines, further improvements are necessary. In this study, we present our endeavor to develop a novel framework called the deep-learning-based photometry and astrometry Bayesian neural network (PNet). PNet includes an advanced architecture that excels in detecting, performing photometry, and carrying out astrometry for point-like celestial objects. Notably, PNet leverages the Bayesian

neural network (BNN) to estimate both the photometry results and their associated uncertainties. The subsequent sections of this paper delve into various aspects of our work. Section 2 discusses the properties of the data and the methods we employed to reprocess the data. In Section 3 we present the structure of PNet. In Section 4 we assess the PNet performance using both simulated data and real observation data. These results are compared to those obtained using SExtractor (Bertin & Arnouts 1996) to show the advantages of PNet. Finally, in Section 5, we present the conclusions of our findings and outline our future research directions.

2. The Data

In this paper, we train and evaluate the performance of our framework using simulated and real observation data. Simulated data allow us to have control over the observation conditions, enabling a more precise assessment of the performance of our framework. On the other hand, real observation data encompass various unknown factors that better reflect the actual performance of our framework. To generate the simulated data, we use Skymaker, a widely used tool for generating synthetic images based on specified observation conditions (Bertin 2009). Skymaker generates simulated images by considering parameters such as PSFs, noise levels, and input star catalogs, which are obtained by the Stuff for galaxies and manually generated catalogs for stars. The distribution of photons that are emitted by celestial objects follows a Poisson distribution, and the PSF serves as the prior distribution function. Additionally, the simulation includes the generation of Poisson-distributed sky-background photons. To account for readout noise, Gaussian noise is simulated, and effects such as blooming or bleeding are also considered in Skymaker. By carefully controlling parameters in the Skymaker, we ensure that the PSF size and noise level closely resemble those of real observation data, thus facilitating a thorough investigation of the performance of our framework.

The real observation data employed in this paper were derived from the Sloan Digital Sky Survey (SDSS) DR17 (Abdurro'uf et al. 2022). The SDSS data were collected using a wide-field 2.5 m telescope (Gunn et al. 2006) located at the Apache Point Observatory in New Mexico (York et al. 2000). These data undergo meticulous processing through a specialized data-processing pipeline (Lupton et al. 2005), which includes precise astrometric calibration (Pier et al. 2003) using the USNO CCD Astrograph Catalog (UCAC; Zacharias et al. 2000), as well as photometric calibration (Padmanabhan et al. 2008) with the aid of numerous standard stars (Smith et al. 2002). Consequently, the data and catalog obtained from the SDSS serve as reliable references for both training and testing purposes. The SDSS employs five filters, namely u , g , r , i , and z filters, which correspond to central wavelengths of 3543, 4770, 6231, 7625, and 9134 Å, respectively.⁴ In this study, we focus on images obtained using the g -band filter due to their relatively higher signal-to-noise ratio. Additionally, the parameters `run`, `camcol`, `field`, and `rerun` define the observation period of the telescope, the camera column number, the area number on the camera, and the version number during data processing, respectively. To ensure the generalization of our algorithm across different sky zones and time periods, we avoid specifying specific values for the first three parameters.

⁴ <https://skyserver.sdss.org/dr1/en/proj/advanced/color/sdssfilters.asp>

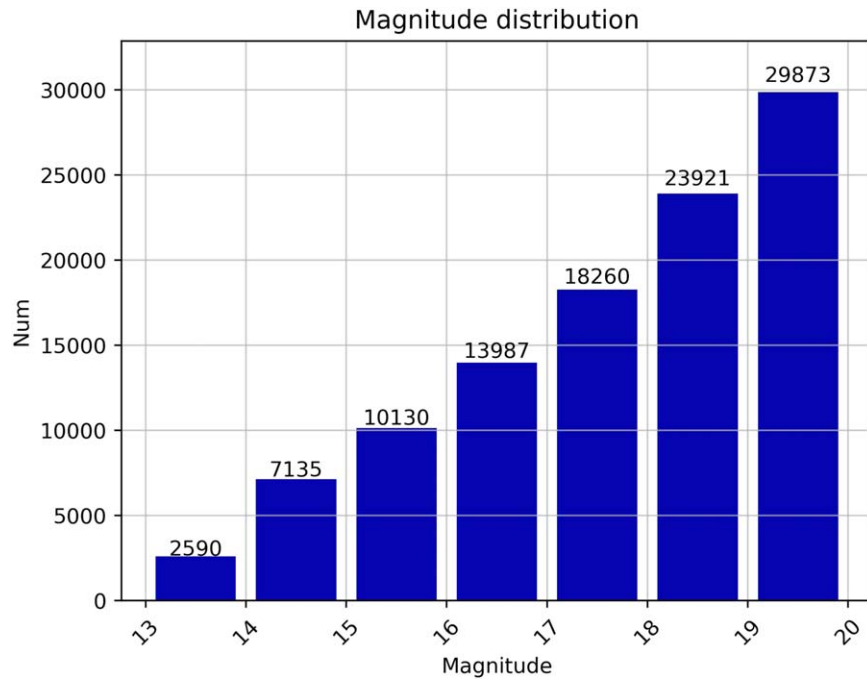


Figure 1. The histogram of stars with different apparent magnitudes in the real data set.

However, we do specify the use of the data-processing process with version number 301 for the rerun parameter.

Given that our framework is designed to specifically detect point-like celestial objects, the data set exclusively consists of such objects. The magnitudes and positions of these point-like celestial objects are determined as regression values within the framework. The resulting positions are provided in camera coordinates, using pixels as a unit of measurement, while the magnitudes are represented as flux f . To calculate the apparent magnitude, we use the equation defined in Stoughton et al. (2002) as follows:

$$\text{mag} = 22.5 - 2.5 \log_{10} f. \quad (1)$$

The apparent magnitude (mag) is related to the flux (f) according to the equation provided. In this study, we adopt a magnitude zeropoint of 22.5 and estimate the magnitudes of stars within the range of 13–20. Figure 1 illustrates the histogram depicting the distribution of stars by different magnitudes in the real data set. The observed distribution aligns with our experience.

Taking into account the impact of the input image size on GPU memory requirements, we divide the original SDSS images into patches with a size of 512×512 pixels. This approach helps reduce the hardware demands. Additionally, we apply certain criteria to remove specific stars that would otherwise necessitate additional processing steps. These criteria include

1. stars located at the image edge within a 10-pixel distance,
2. stars situated near galaxies or other objects, and
3. stars positioned in close proximity to one another within a range of 10 pixels.

3. The Method

The flowchart in Figure 2 illustrates the structure of the proposed framework. Initially, we identify and mask out any

defective pixels present in the original image. Following this, we divide the image into patches with sizes of 512×512 pixels. Subsequently, we employ the photometry-detection net to detect point-like targets within these image patches and conduct photometry to determine the flux of these identified targets. Using the obtained flux values, we perform photometry calibration to derive the magnitudes of the stars. Last, we use the BPNN to reevaluate the magnitudes of the stars and estimate the associated uncertainty in the magnitude measurements. The decision to separate the star detection and the BNN for magnitude measurement is based on the extensive parameterization of DNN, which necessitates multiple sampling during the prediction phase, thereby requiring significant computational resources. In this section, we first introduce several performance evaluation metrics. We then provide a concise overview of the BNN, and we subsequently discuss the implementation details and training procedures of the framework. All the neural networks described in this paper are constructed using PyTorch and executed on a computer equipped with a GTX 3090Ti GPU.

3.1. The Performance Evaluation Criterion

Choosing appropriate performance evaluation criteria is crucial for properly assessing the performance of a framework. It is essential to select evaluation criteria that align with the objectives of developing the framework. Our algorithm focuses on four key aspects: detection of point-like targets, regression of their positions and magnitudes, and estimation of photometric uncertainties. Hence, the following performance evaluation criteria have been selected for our algorithm:

1. The recall rate and the precision rate, and mean average precision (mAP) are chosen to evaluate the performance of target-detection results.
2. The astrometry accuracy in pixels is used to assess the accuracy of the astrometry results.

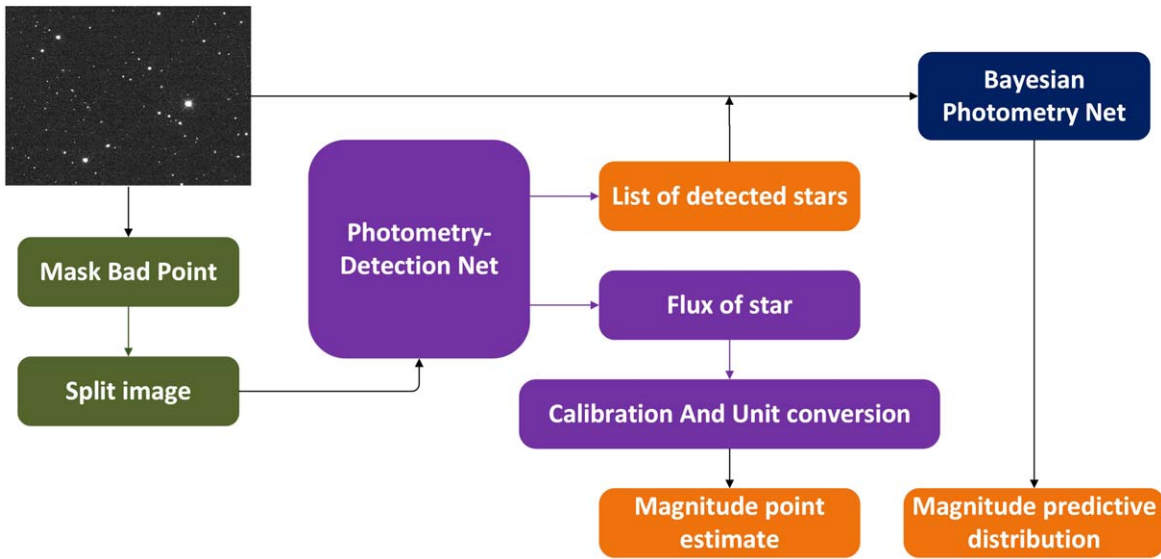


Figure 2. The flowchart of the PNet proposed in this paper. The green block indicates the data-preprocessing part, the purple block indicates the photometry-detection net part, the blue block indicates the BPNN part, and the orange block indicates the output of the whole algorithm.

3. The photometry accuracy in magnitudes is used to evaluate the accuracy of the photometry results.
4. The outlier fraction (η), normalized median absolute deviation (NMAD), median absolute deviation (MAD), and the mean value of the photometry results $1\sigma(\bar{E})$ are used to evaluate the uncertainty of the photometry results.

In the following, we provide a detailed description of the aforementioned performance evaluation criteria. When evaluating the detection results, we consider four possible scenarios:

1. True positive (TP): This occurs when a point-like celestial object is correctly identified as a point-like celestial object.
2. True negative (TN): This occurs when targets other than point-like celestial objects are correctly identified as nonpoint-like celestial objects.
3. False positive (FP): This occurs when targets other than point-like celestial objects are incorrectly identified as point-like celestial objects.
4. False negative (FN): This occurs when point-like celestial objects are incorrectly identified as nonpoint-like celestial objects.

Given that our framework is capable of directly outputting the center coordinates of the point-like celestial objects, we assess the detection results by calculating the Euclidean distance between the predicted position and the corresponding position in the label. The Euclidean distance can be defined using Equation (2),

$$\text{EuclideanDistance} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}, \quad (2)$$

where the variables x_i represent the predicted positions, y_i represent the label positions, and n represents the number of coordinates used to describe the positions (which is 2 in this paper). If the Euclidean distance between the predicted positions and the label positions is below a certain threshold and the classification result is in accordance with the label, it is

considered a TP or TN. Otherwise, it is classified as a FP or FN.

Based on the aforementioned definition, we can assess the performance of our framework in target detection by using the precision and recall metrics, as defined in Equation (3),

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}. \end{aligned} \quad (3)$$

Precision and recall are widely used metrics for evaluating target-detection results. Precision represents the percentage of TPs among all positive detection results, indicating the performance of the detection algorithm in minimizing false alarms. Recall, on the other hand, represents the percentage of TPs among all actual targets, describing the ability of the detection algorithm to identify all positive instances. Precision and recall are both influenced by the chosen detection threshold. A higher threshold leads to a higher precision, but lower recall, and vice versa. To comprehensively evaluate the performance of a detection algorithm, we can vary the detection threshold and generate a precision-recall curve (P-R curve). The area under the P-R curve is known as the average precision (AP). The mean average precision (mAP) is calculated by averaging the AP values across all categories, providing an overall assessment of the performance of the detection algorithm. In this paper, the mAP is obtained using Equation (4), where n represents the number of categories (which is 1 in this case). Furthermore, it is worth noting that the astrometry accuracy can be measured by calculating the distance between the predicted position and the corresponding position in the label for all TP detection results,

$$mAP = \frac{\sum AP}{n}. \quad (4)$$

The BNN is employed to estimate the uncertainty inherent in the data. When a star image is fed into the BNN, it generates a

predictive posterior distribution for the star's magnitude, along with the mean value of that distribution. The mean and standard deviation of this distribution, which cannot be directly obtained, are typically estimated through multiple Monte Carlo sampling iterations. The standard deviation σ_{mag} of this distribution can be interpreted as a quantitative measure of uncertainty. Consequently, we consider two aspects of the output: the deviation of the true value from the mean of the predicted distribution, and the level of uncertainty in the prediction results. To evaluate the deviation of the true value from the mean, we employ the relative error RelativeError and the absolute error δ_{mag} , defined in Equation (5),

$$\begin{aligned} \delta_{\text{mag}} &= \text{mag}_{\text{true}} - \text{mag}_{\text{pred}}, \\ \text{RelativeError} &= \frac{\delta_{\text{mag}}}{1 + \text{mag}_{\text{true}}}. \end{aligned} \quad (5)$$

By considering the relative error for all targets, we can determine the fraction of outliers (η) by establishing a threshold value for the relative error and calculating the proportion of the prediction results that exceeds this threshold (σ_{mag} in this study). Additionally, we employ the normalized median absolute deviation (NMAD) to evaluate the relative errors obtained. The median absolute deviation (MAD) is a statistical measure that characterizes the sample bias of one-dimensional numerical data. To obtain the NMAD, we normalize the MAD by multiplying it by a factor of 1.4826 (Rousseeuw & Croux 1993), as demonstrated in Equation (6),

$$\delta_{\text{NMAD}} = 1.48 \times \text{median}\left(\frac{\text{abs}(\delta_{\text{mag}} - \text{median}(\delta_{\text{mag}}))}{1 + \text{mag}_{\text{true}}}\right). \quad (6)$$

The absolute error is assessed through the mean absolute error (MAE), which represents the average of the absolute differences between the mean of the predictive posterior distribution and the corresponding true value. This metric provides a visual indication of the error level. Regarding the uncertainty, we initially estimate the magnitude distribution for each star using the BNN. We then compute the mean value of these distributions, denoted as \bar{E} , which serves as a reference indicator for the uncertainty distribution.

3.2. The Principles of the Bayesian Photometry Neural Network

The BPNN is used to estimate both the magnitude of stars and the associated photometry uncertainties. In traditional neural networks, the weights are fixed, leading to a lack of uncertainty estimation and excessive confidence in the predicted results. To address this issue, BNNs employ the variational Bayes (VB) method (Blundell et al. 2015) to introduce uncertainty into the network weights. However, before delving into the methods for capturing uncertainty, it is crucial to comprehend the origins of uncertainty.

In the field of machine learning, two main types of uncertainty are commonly recognized: aleatoric uncertainty (AU; also known as data uncertainty), and epistemic uncertainty (EU; also known as model uncertainty; Kiureghian & Ditlevsen 2009). The AU stems from the inherent noise present in the data set itself (Gal et al. 2016). Because this noise is natural and unpredictable, AU cannot be eliminated. On the other hand, the EU arises from insufficient training of the

network, resulting in a lack of knowledge about the system behavior. In principle, this uncertainty can be reduced as the training data approaches infinity (Hora 1996). As mentioned earlier, the predictive uncertainty (PU) we aim to capture can be expressed as the combination of AU and EU (Abdar et al. 2021), as illustrated in Equation (7),

$$\text{PU} = \text{AU} + \text{EU}. \quad (7)$$

By defining the PU, we can establish the underlying principle of the BPNN. Initially, we define the weights of the BPNN as $\omega \in \Omega$, where Ω represents the parameter space of the BPNN. The training data set is denoted as D , and within this data set, we have data pairs X and Y . Similarly, in the test data set, we have data pairs x and y . The distribution of weights ω learned by the network from the data set D is represented as $p(\omega|D)$. Additionally, $p(y|x, \omega)$ signifies the probability that the neural network yields output y when given input x and weight ω . Last, $p(\omega)$ represents the prior weight distribution of the network. With these definitions, we can derive the probability distribution of the output y given the input x when the neural network is trained using data set D , as illustrated in Equation (8),

$$p(y|x, D) = \int p(y|x, \omega)p(\omega|D)d\omega. \quad (8)$$

The prediction uncertainty captured by the BPNN is represented by $p(y|x, D)$. However, obtaining $p(\omega|D)$ through analytical calculations is challenging in real applications, and often requires approximation methods to perform the inference task (Hortúa et al. 2020). In this study, we employ variational inference to approximate the solution for $p(\omega|D)$. Initially, we assume a variational distribution $q(\omega|\theta)$, where θ denotes a set of variational parameters. Subsequently, we calculate the Kullback-Leibler (KL) divergence between the variational distributions $q(\omega|\theta)$ and $p(\omega|D)$. Finally, we determine a set of variational parameters θ^* that minimizes the KL divergence, as shown in Equation (9),

$$\begin{aligned} \theta^* &= \underset{\theta}{\text{argmin}} \text{KL}[q(\omega|\theta) \| p(\omega|D)] \\ &= \underset{\theta}{\text{argmin}} \int q(\omega|\theta) \ln \frac{q(\omega|\theta)}{p(\omega|D)} d\omega. \end{aligned} \quad (9)$$

Because $p(\omega|D)$ cannot be obtained analytically, we further introduce the Bayesian formula below:

$$p(\omega|D) = \frac{p(D|\omega)p(\omega)}{p(D)}. \quad (10)$$

We could obtain the KL divergence according to Equations (9) and (10), as shown in Equation (11),

$$\begin{aligned} \text{KL}[q(\omega|\theta) \| p(\omega|D)] &= \ln p(D) + \text{KL}[q(\omega|\theta) \| p(\omega)] \\ &\quad - \int q(\omega|\theta) \ln p(D|\omega) d\omega. \end{aligned} \quad (11)$$

Because $\ln p(D)$ is only related to properties of the data, we could minimize the KL divergence through minimizing Equation (12),

$$\begin{aligned} F(D, \theta) &= \text{KL}[q(\omega|\theta) \| p(\omega)] - \int q(\omega|\theta) \ln p(D|\omega) d\omega \\ &= E_{q(\omega|\theta)}[\ln q(\omega|\theta) - \ln p(D, \omega)]. \end{aligned} \quad (12)$$

It is important to note that the term $F(D, \theta)$ in Equation (12) corresponds to the negative value of the evidence lower bound

(ELBO), as discussed in Blei et al. (2017). By employing Equation (12), we can transform the analytically challenging calculation into a practical optimization problem for the variational parameter θ . For a comprehensive understanding of the approximation method, we refer to the work of Hortúa et al. (2020), while providing a concise overview here.

To begin, we assume that we can obtain a set of variational parameters $\hat{\theta}$ that minimizes $F(D, \theta)$ through an optimization process. In this study, we perform this optimization using the backpropagation algorithm (Rumelhart et al. 1986) and employ the Adam gradient descent algorithm (Kingma & Ba 2014). With the obtained $\hat{\theta}$, we can derive the predictive distribution $q_{\hat{\theta}}$. By combining this with Equation (8), we can determine the predictive distribution of the output variable y given the input x ,

$$q_{\hat{\theta}}(y|x) = \int p(y|x, \omega) q(\omega|\hat{\theta}) d\omega. \quad (13)$$

Although Equation (13) provides an analytical solution for the predictive distribution, calculating the integral can be challenging in practical applications. Hence, we employ Monte Carlo sampling (Gal et al. 2016) and a series addition method to obtain the final results,

$$q_{\hat{\theta}}(y|x) \approx \frac{1}{N} \sum_{n=1}^N p(y|x, \hat{\omega}_n), \quad (14)$$

where N denotes the number of samples, and $\hat{\omega}_n$ represents the n th sampled value of the weights obtained from $q(\omega|\hat{\theta})$. Equation (14) is equivalent to Equation (13) as the number of samples N tends to infinity. This equation indicates that we could obtain a Bayesian estimation through Monte Carlo sampling of the weights for a predefined neural network.

In the study by Hortúa et al. (2020), the authors use the total variation principle to derive the analytical results for the variance of the predicted distribution on a fixed input x . They further simplify these results to obtain

$$\hat{\text{Var}}(y|x) \approx \frac{1}{T} \sum_{t=1}^T \sigma_t^2 + \frac{1}{T} \sum_{t=1}^T (\mu_t^2 - \bar{\mu}^2), \quad (15)$$

where T denotes the total number of forward passes of the network. The terms σ_t and μ_t refer to the standard deviation and mean of the distribution obtained during the t -th forward pass, respectively, while $\bar{\mu}$ represents the mean of all μ_t values. The first term in Equation (15) corresponds to the AU discussed earlier, whereas the second term corresponds to the EU. In real applications, we build a BPNN with the flipout method discussed in Wen et al. (2018) to generate pseudo-independent weight perturbation on minibatches, which could simulate the Bayesian interference process. Then the distribution between predicted results and prior distribution could be evaluated to provide a reference for the photometry uncertainty. We discuss the details of the method in Section 3.3.2.

3.3. The Structure of Neural Networks in PNet

In this subsection, we provide a comprehensive overview of the neural network structure employed in our framework. The neural network comprises three interconnected components that collaboratively predict the final outcomes. The first component, known as the photometry-detection net, is

designed to detect point-like celestial objects, accurately determine their subpixel positions, and calculate their flux. The second component is responsible for the estimation of the photometry results and their uncertainties. Last, we carry out the photometry calibration, using reference stars to ensure a precise calibration of the photometric measurements derived from the flux values. This framework allows us to obtain reliable magnitudes and positions for point-like celestial objects. It is important to emphasize that the primary focus of this paper is not the detection of various celestial objects such as galaxies or quasars. Neural networks specifically tailored for the detection of celestial objects from multicolor images are more suitable for these targets (Jia et al. 2023b). When these targets have been detected by the methods described above, we proceed to mask them and carry out point-like celestial object detection, astrometry, and photometry using single-band images with our framework. As a result, the final output of our framework includes the positions of point-like celestial objects, their corresponding magnitudes, and the associated photometry uncertainties, which could be directly used to analyze celestial objects with light variation and moving celestial objects.

3.3.1. The Photometry-detection Neural Network

Because point-like celestial objects possess a relatively simple structure and smaller size than other natural images, there is no need to employ highly complex backbone neural networks specifically designed for natural image target detection. Furthermore, our goal is to determine the center of point-like celestial objects instead of using the bounding-box approach commonly used in DNN-based target-detection algorithms. Therefore, we must modify the position regression strategy within the neural networks. With these considerations in mind, we propose a novel structure known as the photometry-detection net, which directly provides the position and flux of point-like celestial objects. Training data for the photometry-detection net consist of original observation images in a single band, and the corresponding labels in the training data indicate the position and flux of these point-like celestial objects.

First and foremost, the photometry-detection net integrates the CenterNet structure for the detection of celestial objects. CenterNet is a straightforward, efficient and accurate neural network designed specifically to detect small targets by regressing their key points (Zhou et al. 2019). CenterNet has gained widespread recognition and has been applied in various domains (Ahmed et al. 2021; Guo et al. 2021). The structure of CenterNet is depicted in Figure 3. In practical scenarios, CenterNet begins by performing regression to identify the center point of a target, followed by feature extraction in the vicinity of the center point. Subsequently, CenterNet provides the target position and confidence score. Building upon the detection results, we introduce the photometry neural network branch connected to CenterNet, enabling us to obtain the flux of the point-like celestial objects. To optimize computational resources, we set the input size of the CenterNet to 512×512 pixels, and the output of the CenterNet consists of the detected targets, their corresponding positions, and their flux values.

As depicted in Figure 3, CenterNet comprises three distinct modules: the downsample module, the extractor module, and the prehead module. In this study, we employ a 2D convolutional neural network as the downsample module, using a convolutional layer with a kernel size of 7×7 , a

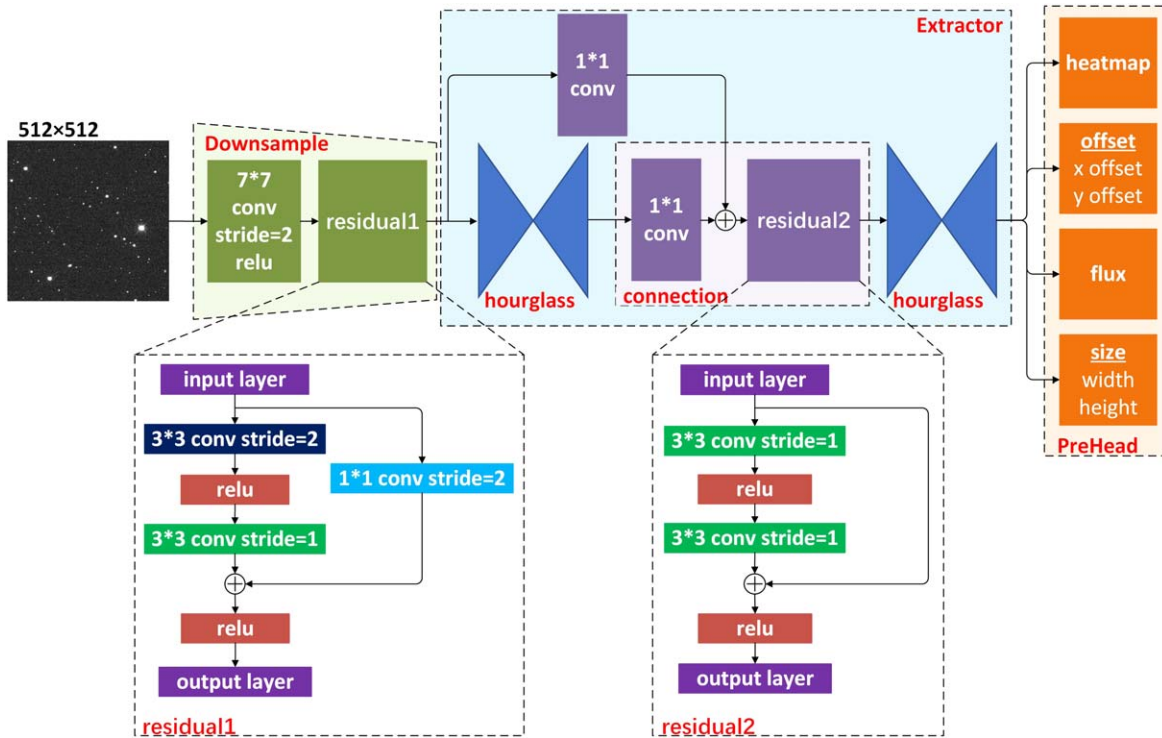


Figure 3. The structure of CenterNet. CenterNet includes three modules: the downsample module, the extractor module, and the prehead module. Residual1 and Residual2 represent two types of residual structures. The network takes an image with a resolution of 512×512 pixels as input and outputs the center position, flux value, size of the target, and a heatmap that corresponds to the input image. The center position and the flux value are used for further processing. In the extractor module, we propose to use the hourglass neural network for feature extraction.

padding of 3, and a stride of 2 to downsample the image to a size of 256×256 pixels. Subsequently, the resulting image is further downsampled to 128×128 pixels using the residual1 component, as illustrated in Figure 3. These downsampled images are then passed through the extractor module to extract their features (Springenberg et al. 2014). For extractor module, we adopt the hourglass neural network in this paper, as it is well suited for capturing features from small objects. When the downsampled images are inputted into the hourglass network, a sequence of residual modules with convolutional layers having a stride of 2 is employed to iteratively downsample the image four times, reducing its size by a factor of two each time. Following downsampling, four upsampling operations are performed using the nearest-neighbor interpolation algorithm. The spatial information from each downsampling scale is preserved through skip connections and fused with the upsampled feature maps during the upsampling process (Newell et al. 2016). The hourglass network ultimately produces a feature map of the same size as the input. By stacking multiple hourglasses, the detection capability of the neural network can be enhanced, as subsequent hourglasses refine the detection results based on the output of the previous ones, which proves more effective than employing a single detection network. For instance, in a system containing multiple stars, a star that has been missed in the initial detection round may become easier to detect in subsequent hourglasses.

Finally, the feature maps extracted from the extractor module are passed to the prehead module to perform regression on various attributes of the targets in the image under processing. In this study, the prehead module consists of four sub-preheads: heatmap, offset, flux, and size. All four preheads are

convolutional neural networks (CNNs) comprising 2D convolutional layers and rectified linear unit (ReLU) layers. The heatmap prehead generates a heatmap of size $\text{batch size} \times \text{class num} \times \text{height} \times \text{width}$. In this particular study, only star detection is considered, so the class num is set to 1. The heatmap divides the original image into a grid of 128×128 patches, serving two purposes: providing confidence scores for different classes at various positions of the target, and indicating the approximate position of the detected targets within the grid. The offset and size preheads predict different properties of the targets, but share the same CNN structure, producing output sizes of $\text{batch size} \times 2 \times \text{width} \times \text{height}$, where 2 represents the two parameters predicted by each prehead. The offset prehead estimates the deviation between the actual center position of the target and the position indicated by the heatmap, which allows us to calculate the actual center position. The size prehead provides the height and width of the target. The flux prehead has a structure similar to the offset prehead, but with a reduced number of output channels (1) and prediction of a single parameter, which represents the flux. The structures of the four preheads are depicted in Figure 4. Additionally, it is worth noting that we have incorporated several intermediate supervision steps within the prehead module. These supervisions are added directly after each hourglass neural network to assess its performance during the training phase. After training, these neural networks are not used, and we solely employ the main structure of the prehead module.

3.3.2. The Bayesian Photometry Neural Network

The BPNN is employed to determine the magnitude and uncertainty of photometry results, serving as a reference for subsequent light-curve classification. Based on the detection results from the photometry-detection net, all detection results

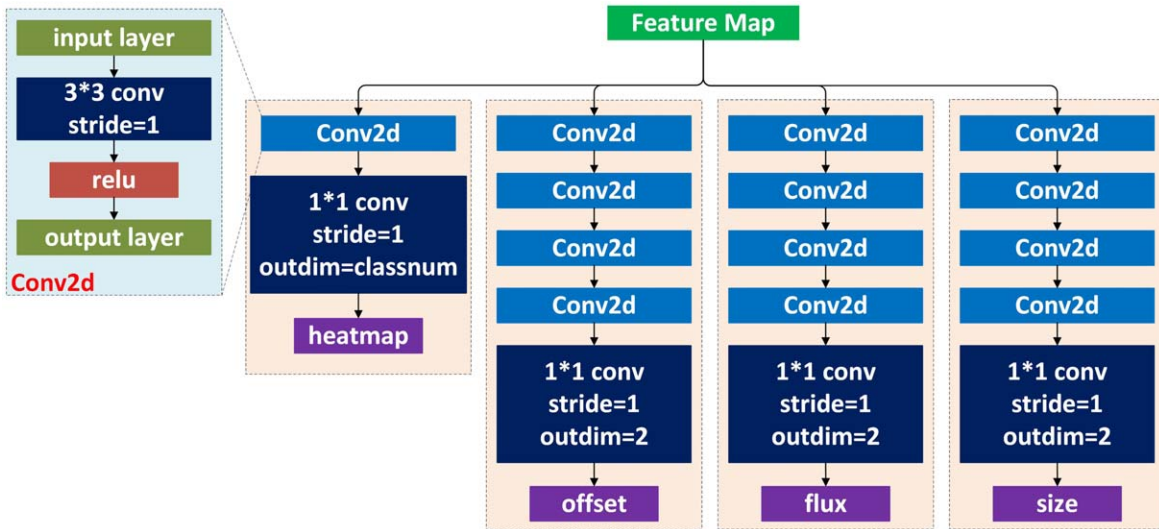


Figure 4. Structure diagram of the preheads. The blue blocks share the same structure, and their detailed configuration is presented in the figure. The yellow 1×1 convolution within the prehead module is used to adjust the number of channels to align with the desired output parameter count. The input to the prehead is the feature map extracted by the hourglass, while the output consists of the parameters indicated in the purple blocks of the figure.

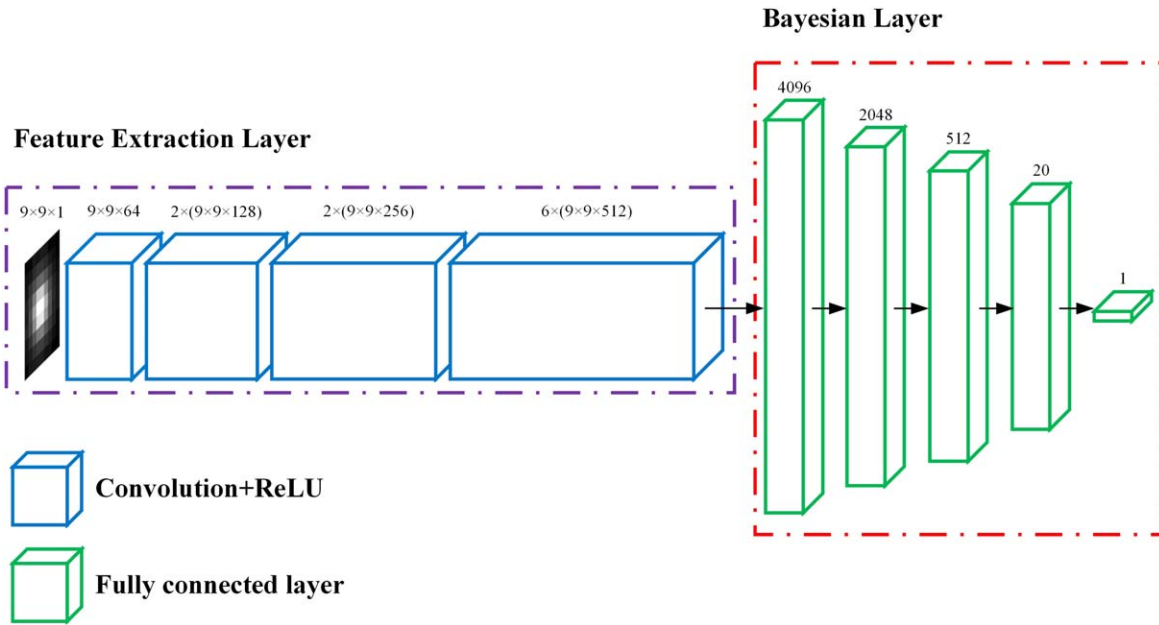


Figure 5. Architecture of the BPNN. The feature-extraction layer is depicted in purple, and the Bayesian layer is represented in red. The blue blocks indicate the convolutional layer, and the green blocks correspond to the Bayesian fully connected layer. The network takes a star image of dimensions 9×9 pixels as input and produces a predicted distribution for photometry as the output.

are cropped into stamp images with a size of 9×9 pixels. This size is suitable for star images of moderate brightness, but the stamp images can be adjusted to smaller or larger sizes depending on the actual observation conditions. The stamp images are then used as input for the BPNN, which estimates the magnitude multiple times using the flipout method to approximate the posterior distribution of the neural networks in magnitude estimation. The BPNN comprises two main components: the feature-extraction layer, and the Bayesian layer, as illustrated in Figure 5.

The feature-extraction layer is built upon the resnet50 architecture (He et al. 2016). Resnet50 tackles the challenges of increased computational time and diminished accuracy in deep

network structures by incorporating shortcut connections. Because the input image is relatively small (9×9 pixels), we have adjusted the size of the convolution kernel to 3×3 to better accommodate these small-scale images. Additionally, we removed the pooling layers in the network to reduce information loss and improve prediction accuracy. Last, we eliminated the fully connected layer and repurposed it as the feature-extraction layer.

In Section 3.2 we have demonstrated that the variational Bayesian (VB) method can approximate the posterior distribution by minimizing the evidence lower bound (ELBO), as depicted in Equation (12). Typically, this procedure is accomplished using the gradient descent method. The gradient

of $F(D, \theta)$ in Equation (12) can be computed by considering the density function $q(\omega|\theta)$, which is also parameterized by θ ,

$$\nabla_{\theta} E_{q(\omega|\theta)}[f_{\theta}(\omega)] = \int \nabla_{\theta} q(\omega|\theta) f_{\theta}(\omega) d\omega + E_{q(\omega|\theta)}[\nabla_{\theta} f_{\theta}(\omega)]. \quad (16)$$

The specific form of $f_{\theta}(\omega)$ is

$$f_{\theta}(\omega) = \ln q(\omega|\theta) - \ln p(D, \omega). \quad (17)$$

As depicted in the first term of Equation (16), computing the gradient necessitates obtaining analytical solutions for expectations involving the approximate posterior distribution. However, this task often proves challenging in real-world applications. Consequently, when attempting to directly compute Equation (12) using a neural network via forward propagation, calculating the gradient becomes generally infeasible, and the computation process lacks differentiability, impeding backpropagation. To address this issue, the so-called reparameterization trick (RT) was introduced by Kingma & Welling (2013). By employing this trick, a straightforward and differentiable unbiased estimator for the ELBO can be generated, enabling the use of gradient descent algorithms for ELBO optimization.

However, the RT is not without limitations. One issue arises from the fact that all sample weights within the same batch are identical, resulting in correlated gradients across different examples in the batch (Hortúa et al. 2020). To address this problem, a technique known as flipout was introduced by Wen et al. (2018). Flipout facilitates the generation of efficient pseudo-independent weight perturbations on minibatches. In a comparative study conducted by Hortúa et al. (2020), several methods, including dropout, dropconnect, RT, and flipout, have been evaluated, with flipout demonstrating superior performance. As a result, for the implementation of the BNN in the BPNN, we opted to use flipout. Specifically, we employ the method proposed in Krishnan et al. (2022) to construct a five-layer flipout linear layer, thereby enabling the realization of Bayesian layers. In the flipout linear layers, multiple sets of weights are sampled during each forward pass. During the training step, the weights are randomly flipped or flipped out based on a Bernoulli distribution. In the deployment step, these weights are treated as random variables and can be used for the uncertainty estimation. By sampling multiple sets of weights, flipout introduces stochasticity into the network, resulting in different predictions for the same input. This characteristic allows for the quantification of the uncertainty and enhances the model robustness and generalization capabilities.

3.3.3. The Photometry Calibration Part

With the above neural networks, we are able to estimate the flux of celestial objects based solely on the grayscale counts within the images of these objects. However, in order to identify flares and variable stars, it becomes crucial to calibrate the flux of all detected targets by referencing the flux from reference stars. For this calibration process, we employ the method proposed by Stoughton et al. (2002) to convert flux into magnitudes, which is represented by Equation (18),

$$\text{mag} = \text{mag}_0 + \text{mag}_{\text{zero}} + k(i) \times x + f(i). \quad (18)$$

The equation calculates the calibrated magnitude mag based on various variables. The instrumental magnitude mag_0 is derived from the photometry branch, and mag_{zero} represents the zeropoint magnitude set by the user. The primary extinction

coefficient is denoted as k , and x represents the airmass. Additionally, $f(i)$ characterizes the flat field of the CCD i th column. Consequently, when working with SDSS data, it is necessary to obtain the run, camcol, and field parameters for the magnitude calibration. On the other hand, for data obtained by different telescopes, we need to first use the PNet to obtain flux and detection results. Then, we use 50–100 isolated reference stars on the same image with known magnitudes to fit the calibration function by either least squares or minimizing the chi-square distance between the flux and the magnitudes. In the above steps, we filter out abnormal points where the error between the observed value and the expected value exceeds 1σ .

4. Training and Performance Evaluation

In this chapter, we use two types of data to train and evaluate the performance of our framework: the g-band data from SDSS DR17, and simulated data generated by Skymaker. It is important to highlight that while the data sets differ, all other computational procedures in the framework remain consistent. In addition, we select SExtractor (Bertin & Arnouts 1996), a commonly used astrometry and photometry tool for data processing, to process the observation data at the same time for comparison. As a benchmark, we employ the magnitudes and coordinates provided by the official catalog from the SDSS for real observation data and catalog provided by the Skymaker for simulated data. Regarding the photometry-detection net, we show the detection results as well as preliminary photometry results. Subsequently, we focus on the BPNN and present the estimation of the photometry uncertainty.

4.1. Training and Performance Evaluation of the Photometry-detection Net

During the training phase, certain targets that are not used in the training data, such as dense star fields and stars close to galaxies, are masked. As previously mentioned, the detection of these targets from multiple color images requires specialized processing methods, which are beyond the scope of this paper. For optimization, we employ the Adam optimizer (Kingma & Ba 2014) and evaluate the classification results using the focal loss (Lin et al. 2017). The focal loss dynamically adjusts the impact of each training example on the overall loss, ensuring a balanced approach to the detection results of celestial objects with varying magnitudes. The focal loss is formulated by introducing a modulating factor into the cross-entropy (CE) loss, as originally proposed by Lin et al. (2017). In this study, we employ an alpha-balanced version of the focal loss, as defined in Equation (19),

$$\text{FocalLoss}(p_t) = -\alpha_t(1 - p_t)^{\gamma} \log(p_t), \quad (19)$$

where α_t is the weighting factor used to adjust the weights of positive and negative samples, p_t is the output of the network, and $\gamma > 0$ is an adjustable focusing parameter (2 in this paper). The mean absolute error (MAE) loss is used to assess the position error, while the mean square error (MSE) loss is employed to evaluate flux prediction outcomes. The computation of MSE and MAE is demonstrated in Equation (20), where n represents the total number of targets, \hat{y}_i denotes the predicted value of the i th target, and y_i denotes the corresponding true value. The aggregate of these three losses constitutes the loss function employed to train the photometry-detection net, as

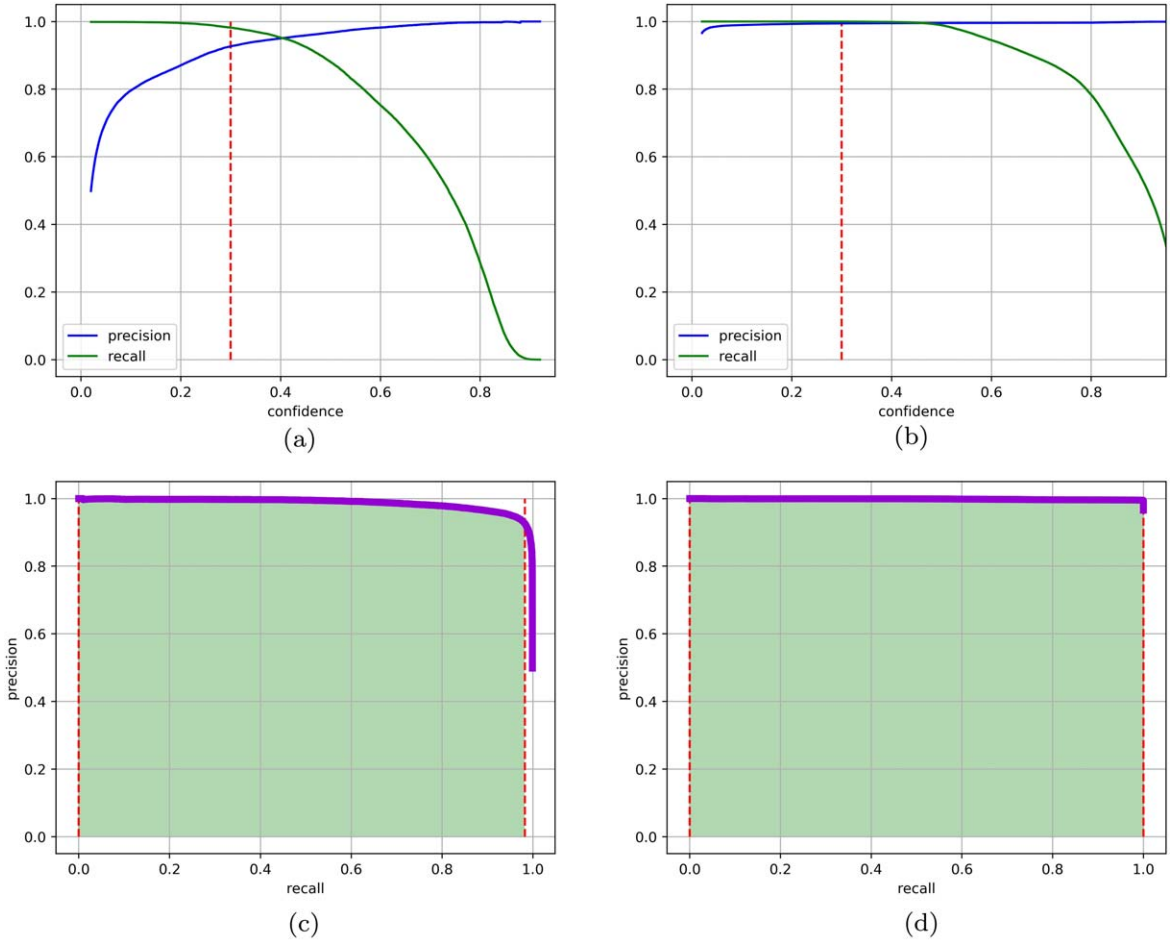


Figure 6. In panels a and b, the horizontal axis represents the confidence level of the predicted targets by the photometry-detection net, and the vertical axis represents the corresponding precision and recall at that confidence level. The dashed red line represents the confidence threshold of 0.3 that was used in this study. In panels c and d, the horizontal axis represents recall, and the vertical axis represents precision. The purple curve illustrates the precision-recall (p - r) curve when the confidence threshold is set to 0. The area between the two dashed red lines depicts the p - r curve when the confidence threshold is set to 0.3, and the region between this curve and the horizontal axis is visually represented as the green area in the figure. This provides insights into the photometry-detection net performance in detecting star targets. As explained in this section, the area between the p - r curve and the horizontal axis corresponds to the AP. A higher AP value indicates a better target-detection performance by the network. As shown in this figure, our method could obtain an AP more than 98% in simulated and in real observation data. (a) The relation between the confidence and the precision and the recall for the photometry-detection net in the SDSS data. (b) The relation between the confidence and the precision and the recall for the photometry-detection net in the simulated data. (c) The precision-recall curve for the photometry-detection net in the SDSS data. (d) The precision-recall curve for the photometry-detection net in the simulated data.

depicted in Equation (20),

$$\begin{aligned} \text{MAE} &= \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \\ \text{MSE} &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2. \end{aligned} \quad (20)$$

It takes approximately 324 s to complete a single epoch when training the photometry-detection net on a computer equipped with a 3090Ti GPU. The batch size consists of 40 images, each with dimensions of 512×512 pixels. After approximately 40 to 50 epochs of training with randomly initialized weights, the photometry-detection net starts to converge. The convergence process is accelerated when using pretrained weights. Because the instrumental magnitude measurements exhibit a relatively consistent pattern, when employing pretrained weights and training on new data batches, it is common practice to freeze the feature-extraction network (the hourglass network) and the photometry branch network. After training for a specific

	SExtractor_SDSS	PNet_SDSS	PNet_Simulation
Precision	84.24%	92.64%	99.43%
Recall	96.12%	98.20%	99.98%

number of epochs, these two networks are then unfrozen, and the overall network undergoes fine-tuning.

After training, we have tested the performance of photometry-detection net in detecting star targets and measuring magnitudes using a batch of 512×512 pixel SDSS astronomical images and simulated images generated by Skymaker. When the network outputs classification results, it provides a confidence level, and when this level is higher than a certain threshold, we consider that there exist star targets. Generally, the lower the threshold, the higher the recall, but the lower the precision, and vice versa. We show the performance of photometry-detection net using different confidence thresholds

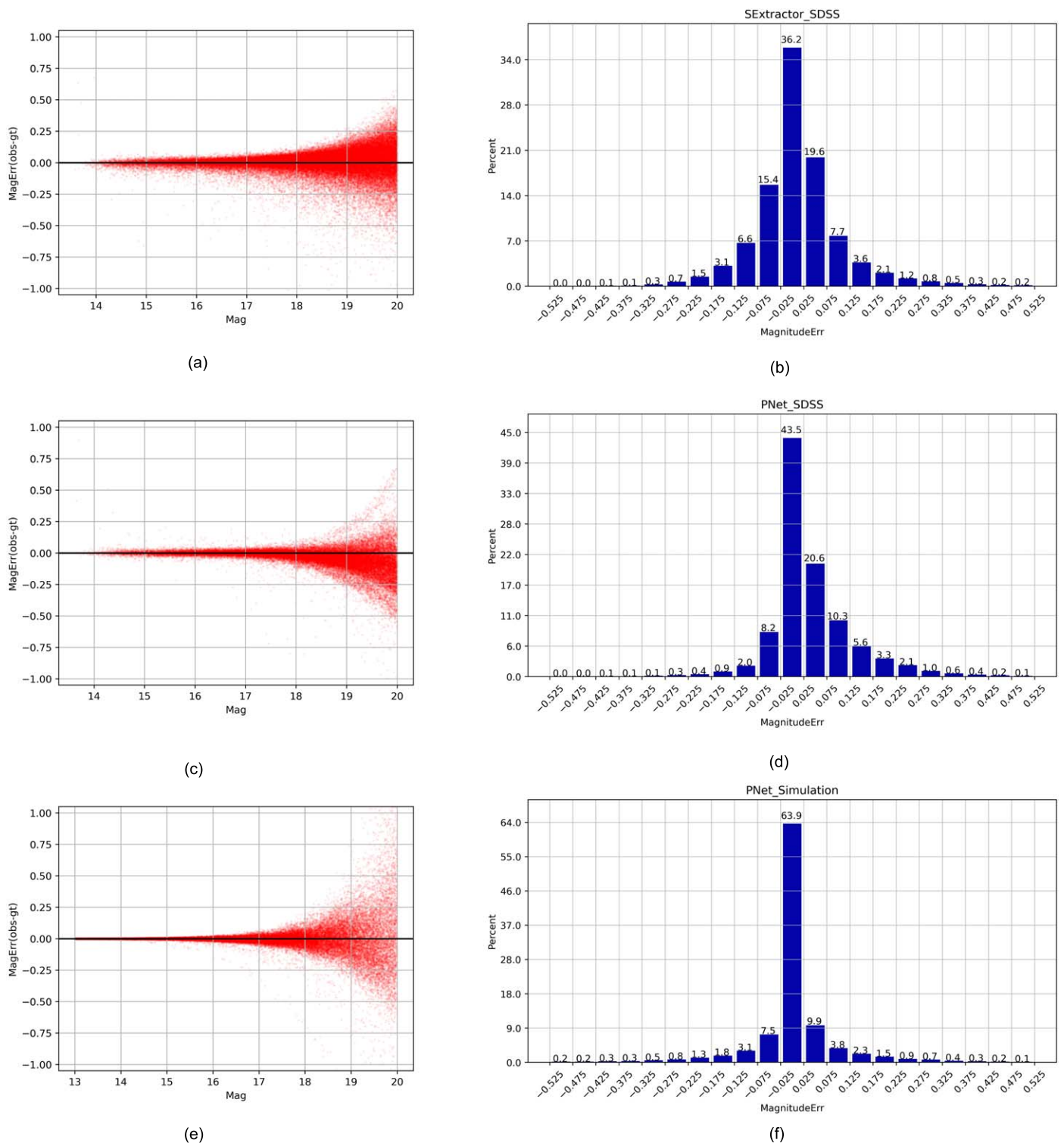


Figure 7. Panels (a), (c), and (e) display the photometric error of SExtractor on SDSS data, the photometry detection on SDSS data, and the photometry detection on simulated data, respectively. In the top panel, the x -axis represents the magnitude, and the y -axis represents the error between the predicted and true magnitudes. The proximity of the scatter points to the $y = 0$ line indicates the accuracy of the network photometry results. The figure illustrates that our algorithm exhibits smaller errors in regions with lower magnitudes compared to those with higher magnitudes. Panels b, d, and f display the histograms of the magnitude errors. The horizontal axis signifies the magnitude error range, and the vertical axis indicates the percentage of stars falling into each error range. A higher concentration of stars within smaller magnitude error ranges on the horizontal axis indicates a more robust capability of the algorithm to measure target magnitudes. As depicted in these panels, PNet exhibits superior photometry results compared to those of SExtractor. (a) The photometry error of SExtractor for the SDSS data. (b) Histogram of the photometry error of SExtractor for the SDSS data. (c) The photometry error of PNet for the SDSS Data. (d) Histogram of the photometry error of PNet for the SDSS data. (e) The photometry error of PNet for the simulated data. (f) Histogram of the photometry error of PNet for the simulated data.

in Figure 6. When the confidence threshold is set to 0.3, the specific results obtained from testing on SDSS data and simulated data are shown in Table 1. The table also presents the

detection results from SExtractor. It is evident that our method achieves a higher level of precision and recall rate than SExtractor.

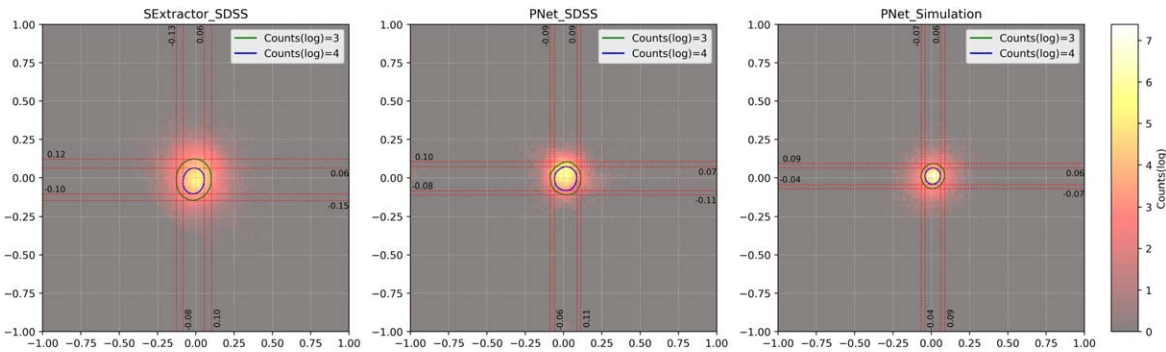


Figure 8. The left panel displays the astrometry error of SExtractor on the SDSS data, the middle panel illustrates the astrometry error of the photometry-detection net on the simulated data, and the right panel shows the astrometry error of the photometry-detection net on the simulated data. Assuming the true positions of all targets are centered at (0,0), the predicted positions are scattered across the figure. Each small area within the figure, defined as a 0.02×0.02 pixel region, tallies the number of stars with astrometry errors falling within that area. The resulting count is logarithmically transformed and visualized as a heatmap. Both the horizontal and vertical axes are measured in pixels, while the color bar denotes the logarithm of the star count. The blue contour corresponds to an astrometry error of 0.15 pixels, the green contour corresponds to an astrometry error of 0.21 pixels, and the dashed red line represents the limit of the astrometry error in both the x and y directions. When the centers of the green and blue contour lines are closer to the (0,0) point and the contours are smaller and more symmetrical, it indicates improved astrometry accuracy in the algorithm.

During the star target-detection process, photometry-detection net also conducts photometry and astrometry. For the SDSS data, the photometry branch uses the calibration method described in Section 3.3.3 to calibrate the magnitudes. However, because the simulated data do not account for airmass effects or other effects, calibration is not necessary for these data. When using SExtractor to process SDSS data, a fixed offset between the photometry and astrometry results and their ground-truth values may exist. This offset can be determined by statistically analyzing the measurement results. In our study, we adjusted the results obtained from SExtractor by subtracting this offset. Except for this adjustment, no further processing has been performed on the measurement results derived from SExtractor for the SDSS data. The photometry and astrometry results are depicted in Figures 7 and 8.

Figure 7 reveals that the magnitude error is mostly within 0.5 mag for the majority of stars, with approximately 50% of the stars exhibiting magnitude measurement errors within 2.5%. These measurement errors tend to increase gradually as the magnitude of the stars increases. This behavior aligns with theoretical analysis, as the impact of noise with the same level has a smaller effect on stars with lower magnitudes. For every 5 mag decrease, the flux of the star increases by a factor of 100. Additionally, compared to the errors observed in the SDSS data, the measurement errors of the stellar magnitudes in the simulated data are generally smaller and more concentrated. This difference primarily stems from the presence of noise in real data, which not only interferes with the flux measurements of the algorithm, but also affects the process of calibrating the stellar magnitudes. Additionally, as illustrated in Figure 7, it is worth noting that the photometric error of our framework when applied to SDSS data exhibits some asymmetry, primarily due to the calibration process. However, in the overall performance, our framework shows more consistent and reliable photometric results that outperform those obtained by SExtractor on the SDSS data.

Meanwhile, we compared the astrometry results obtained by the photometry-detection net and SExtractor in Figure 8. The astrometry error manifests as a roughly symmetrical circle. The contour lines marked with values 3 represent a diameter of 0.15 pixels, while those with values 4 represent a diameter of 0.21 pixels. In contrast to SExtractor, our framework

demonstrates superior astrometry accuracy when applied to the SDSS data, with a higher number of stars exhibiting an astrometry error smaller than 0.1. Nevertheless, it is worth noting that the astrometry error for the SDSS data remains somewhat larger than that observed in the simulated data, primarily due to the introduction of other noise during real observations.

4.2. Training and Performance Evaluation of the Bayesian Photometry Neural Network

As mentioned earlier, the detected results in the images are segmented into smaller patches to estimate the photometry results and uncertainties. We assume a Gaussian distribution as the prior distribution for the photometry results, and the loss function can be directly derived from Equation (12). For optimization during the training stage, we use the Adam optimizer (Kingma & Ba 2014). However, during the training stage, there is a possibility of encountering the issue of exploding gradients due to the random weight sampling, which can lead to unstable training stage. To address this problem, we implement gradient clipping by setting a threshold. If the gradient surpasses this threshold, it is truncated to a specific value. On average, training one epoch typically takes approximately 12 s for a batch of 2000 small images.

Upon completing the training phase, we proceed to assess the performance of the BPNN using a data set of 14,000 target samples. To obtain the distribution of estimated magnitudes for each star target, we employ Monte Carlo sampling with 50 discrete samples. It is worth noting that due to the large number of parameters involved and the challenges inherent in optimizing BNNs, the estimated uncertainty distribution may not closely align with the true distribution. In order to address this, we propose using the method introduced by Chung et al. (2021) to calibrate the predicted uncertainty results. This calibration algorithm, based on isotonic regression (Kuleshov et al. 2018), adjusts the uncertainty represented by the standard deviation by determining an appropriate calibration coefficient. After measuring the magnitudes of all star targets, the BPNN provides corresponding mean values and standard deviations to characterize the uncertainty distribution. We employ

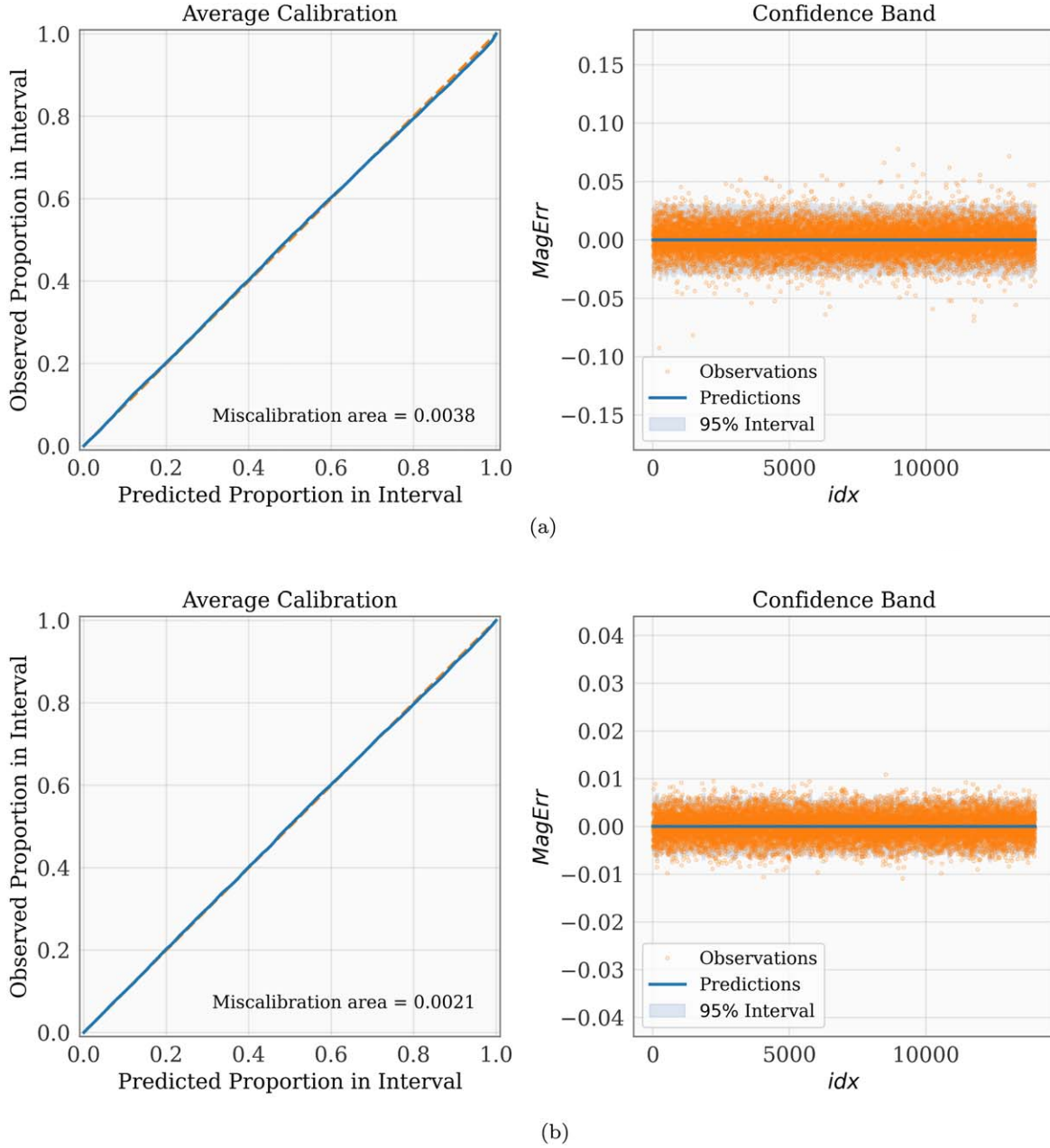


Figure 9. Panel a displays the outcomes of the BPNN in measuring the photometry of targets within the SDSS data, and panel b shows the results for simulated data. In the “Average Calibration” plot (Tran et al. 2020), the horizontal axis represents the expected proportion, while the vertical axis represents the actual observed proportion. The “Miscalibration area” pertains to the space between the curve and the dashed diagonal line with a slope of one. A lower value indicates more accurate uncertainty calibration results. In the “Confidence Band” plot, the horizontal axis corresponds to the target index, and the vertical axis represents the error between the mean of the target distribution calculated by the BPNN and the true value. To align all mean values of the target distributions with zero, we shift them accordingly, as indicated by the blue line at $y = 0$ within the figure. The orange dots illustrate the discrepancy between the true values and the mean of the target distribution, and the blue region denotes the 95% confidence interval projected by the BPNN. A favorable outcome in uncertainty prediction occurs when the light blue region encompasses the orange dots. (a) SDSS photometry results and uncertainties obtained by the BNN. (b) Simulated photometry results and uncertainties obtained by the BNN.

Equation (21) to standardize these distributions,

$$\text{NormalDistribution} = \frac{y_{\text{true}} - \mu}{\sigma}. \quad (21)$$

Here, μ represents the mean of the predicted distribution, y_{true} represents the true value of the corresponding target, and σ denotes the standard deviation of the predicted distribution. By using the aforementioned method, we are able to obtain the distribution of all targets on a standardized evaluation scale. Subsequently, we compare the standardized distribution of

magnitudes for each target with a standard normal distribution having a mean of 0 and a standard deviation of 1. This comparison allows us to assess the accuracy of our Bayesian network quantification of uncertainty. The results are depicted in Figure 9.

More specifically, we use the standard normal distribution as the probability density function (PDF) to conduct a symmetrical search for boundaries from the center outward. This search aims to identify the boundaries at which the ratio of the integral of the PDF within the boundary to the infinite integral

Table 2
Uncertainty of the Photometry Measurement Results on SDSS and Simulated Data

	SDSS	Simulation
η	0.3143%	0.1879%
σ_{NMAD}	6.047×10^{-4}	1.573×10^{-4}
MAE	9.697×10^{-3}	2.105×10^{-3}
\bar{E}	7.004×10^{-2}	1.682×10^{-2}

of the PDF equals the horizontal axis value in the average calibration plot of Figure 9. The vertical axis represents the ratio of the number of standardized targets within the observed boundary to the total number of targets. Because the PDF follows a standard normal distribution, the infinite integral value is 1. In an ideal scenario, in which the uncertainty can be accurately quantified, these two ratios should align, resulting in a diagonal line with a slope of 1 in the average calibration plot of Figure 9. When the curve bends downward, indicating that the predicted proportion in the interval exceeds the observed proportion in the interval, it signifies an overconfident state of the uncertainty estimation (where the predicted uncertainty is too small). Conversely, when the curve bends upward, indicating that the predicted proportion in the interval falls below the observed proportion in the interval, it suggests an underconfident state of the uncertainty estimation (where the predicted uncertainty is too large).

Overall, the BPNN has the ability to predict the uncertainty distribution of magnitude estimation results. When we have obtained the magnitude distribution predictions for all targets using this network, we can assess these predictions using the evaluation criterion described in Section 3.1. By defining the boundary of the relative error as positive and negative $3\sigma_{\text{relative}}$, we can examine the results presented in Table 2. It is worth noting that the calculation method for σ_{relative} involves initially calculating the relative error of all targets using the procedure outlined in Equation (6), and subsequently determining the standard deviation of these relative errors as σ_{relative} . This calculation method differs from the σ value that represents the uncertainty computed from the distribution of an individual target. Based on the data presented in Table 2, it becomes apparent that the outlier fraction obtained from the simulated data is lower than that from the real data, suggesting that the algorithm demonstrates greater stability when applied to simulated data. Furthermore, the σ_{NMAD} and MAE values, assessed for both simulated and real data, indicate that the dispersion of predictions from this algorithm is lower, resulting in overall smaller errors when dealing with simulated data. Additionally, the parameter \bar{E} indicates that the BPNN expresses higher confidence in magnitude measurements for simulated data. In general, the results obtained from simulated data exhibit higher accuracy and confidence compared to real data. This discrepancy could potentially be attributed to the effects induced by noise present in real observational data. On one hand, the presence of noisy pixels can interfere with the Resnet50 model extraction of image features, particularly when stars have low signal-to-noise ratios, intensifying such interference. On the other hand, noise can also disrupt the statistical photometry results obtained by the Bayesian layer when leveraging image information. This interference manifests as

an increased uncertainty in the output results, which indicates a lower reliability and credibility.

5. Conclusions and Future Works

In this paper, we introduce PNet, a novel approach for star detection, photometry, and estimation of photometry uncertainties. By leveraging the Bayesian photometry network and the photometry-detection net, PNet offers a comprehensive solution for photometry and astrometry of point-like celestial objects. To evaluate its performance, we conduct tests using both SDSS data and simulated data. The results indicate that our algorithm achieves consistent and reliable results in the simulated data. However, when applied to real data, the presence of noise or undisclosed data-processing steps may introduce certain errors. Nonetheless, the overall results are deemed satisfactory.

There are several additional points that need to be addressed. First, it is crucial to investigate and adopt data-preprocessing methods proposed by other teams for the magnitude and position calibration. Given the prevalent use of CMOS cameras, which differ significantly from CCD cameras, we must also explore suitable data-preprocessing approaches specific to CMOS cameras. Furthermore, it is important to acknowledge the rapid advancements in neural networks in recent years. Exploring alternative methods such as a neural network search or meta-learning could potentially yield improved neural network architectures. Last, integrating the results obtained from the BPNN with the light-curve classification algorithm is essential for the development of new techniques in transient discovery. These techniques will prove valuable for future sky-survey projects, such as the Large Synoptic Survey Telescope (LSST; Ivezić et al. 2019), the Chinese Space Station Telescope (CSST; Gong et al. 2019), and the SiTian Project (Liu et al. 2021).

Acknowledgments

First, we express our gratitude to the referee, whose valuable feedback and guidance over the course of more than two years have significantly contributed to the improvement of our method. Peng Jia would like to thank Professor Zhaohui Shang from the National Astronomical Observatories, Professor Jian Ge from Shanghai Astronomical Observatory, Professor Rongyu Sun from Purple Mountain Observatory, Professor Huigen Liu from Nanjing University, and Professors Chengyuan Li and Bo Ma from Sun Yat-Sen University, who provided very helpful suggestions for this paper. Furthermore, we would like to announce that the code used in this article will be made available in the PaperData repository, which is powered by China-VO, ensuring easy access for interested researchers.

Furthermore, we express our gratitude for the generous financial support provided by the National Natural Science Foundation of China (NSFC) under grant Nos. 12173027 and 12173062, as well as the Major Key Project of PCL. We also acknowledge the science research grants received from the China Manned Space Project with No. CMS-CSST-2021-A01 and the Square Kilometre Array (SKA) Project with NO. 2020SKA0110102. Additionally, we extend our appreciation to the Civil Aerospace Technology Research Project (D050105) and the French National Research Agency (ANR) for their

support in the form of the ANR APPLY grant (ANR-19-CE31-0011) coordinated by B. Neichel.

Funding for the Sloan Digital Sky Survey IV has been provided by the Alfred P. Sloan Foundation, the U.S. Department of Energy Office of Science, and the Participating Institutions. SDSS-IV acknowledges support and resources from the Center for High Performance Computing at the University of Utah. The SDSS website is www.sdss4.org. SDSS-IV is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS Collaboration including the Brazilian Participation Group, the Carnegie Institution for Science, Carnegie Mellon University, Center for Astrophysics | Harvard & Smithsonian, the Chilean Participation Group, the French Participation Group, Instituto de Astrofísica de Canarias, The Johns Hopkins University, Kavli Institute for the Physics and Mathematics of the Universe (IPMU)/University of Tokyo, the Korean Participation Group, Lawrence Berkeley National Laboratory, Leibniz Institut für Astrophysik Potsdam (AIP), Max-Planck-Institut für Astronomie (MPIA Heidelberg), Max-Planck-Institut für Astrophysik (MPA Garching), Max-Planck-Institut für Extraterrestrische Physik (MPE), National Astronomical Observatories of China, New Mexico State University, New York University, University of Notre Dame, Observatório Nacional/MCTI, The Ohio State University, Pennsylvania State University, Shanghai Astronomical Observatory, United Kingdom Participation Group, Universidad Nacional Autónoma de México, University of Arizona, University of Colorado Boulder, University of Oxford, University of Portsmouth, University of Utah, University of Virginia, University of Washington, University of Wisconsin, Vanderbilt University, and Yale University.

Software: Skymaker (Bertin 2009), Bayesian-Torch (Krishnan et al. 2022), Uncertainty Toolbox (Chung et al. 2021), PyTorch (Paszke et al. 2019), Astropy (Robitaille et al. 2013), Matplotlib (Hunter 2007), Numpy (Harris et al. 2020), Scipy (Virtanen et al. 2020), Pandas (McKinney et al. 2011), tqdm (da Costa-Luis 2019), photutils (Bradley et al. 2016), OpenCV (Bradski & Kaehler 2008), Pillow (Lundh & Contributors 1995-2021).

ORCID iDs

Peng Jia  <https://orcid.org/0000-0001-6623-0931>

References

- Abdar, M., Pourpanah, F., Hussain, S., et al. 2021, *Inf. Fusion*, 76, 243
- Abdurro'uf, Accetta, K., Aerts, C., et al. 2022, *ApJS*, 259, 35
- Agarwal, D., Aggarwal, K., Burke-Spolaor, S., Lorimer, D. R., & Garver-Daniels, N. 2020, *MNRAS*, 497, 1661
- Ahmed, I., Ahmad, M., Rodrigues, J. J., & Jeon, G. 2021, *Appl. Soft Comput.*, 107, 107489
- Andrew, Y. 2023, arXiv:2305.00002
- Bertin, E. 2009, *Memorie della Società Astronomica Italiana*, 80, 422
- Bertin, E., & Arnouts, S. 1996, *A&AS*, 117, 393
- Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. 2017, *JASA*, 112, 859
- Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. 2015, in *Proceedings of Machine Learning Research*, 37, *Proceedings of the 32nd Int. Conf. on Machine Learning*, ed. F. Bach & D. Blei (Lille, France: PMLR), 1613, <https://proceedings.mlr.press/v37/blundell15.html>
- Boucaud, A., Huertas-Company, M., Heneka, C., et al. 2020, *MNRAS*, 491, 2481
- Bradley, L., Sipocz, B., Robitaille, T., et al., 2016 Photutils: Photometry tools, *Astrophysics Source Code Library*, ascl:1609.011
- Bradski, G., & Kaehler, A. 2008, *Learning OpenCV: Computer vision with the OpenCV Library* (Sebastopol, CA: O'Reilly Media, Inc.)
- Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., & Maureira, J.-C. 2017, *ApJ*, 836, 97
- Casetti-Dinescu, D. I., Girard, T. M., Baena-Gallé, R., Martone, M., & Schwendemann, K. 2023, *PASP*, 135, 054501
- Cheng, T.-Y., Conselice, C. J., Aragón-Salamanca, A., et al. 2021, *MNRAS*, 507, 4425
- Chung, Y., Char, I., Guo, H., Schneider, J., & Neiswanger, W. 2021, arXiv:2109.10254
- da Costa-Luis, C. O. 2019, *JOSS*, 4, 1277
- Domínguez Sánchez, H., Margalef, B., Bernardi, M., & Huertas-Company, M. 2022, *MNRAS*, 509, 4024
- Duev, D. A., Mahabal, A., Masci, F. J., et al. 2019, *MNRAS*, 489, 3582
- Fariás, H., Ortiz, D., Damke, G., Arancibia, M. J., & Solar, M. 2020, *A&C*, 33, 100420
- Gal, Y., et al. 2016
- Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. 2021, arXiv:2107.08430
- Gómez, C., Neira, M., Hernández Hoyos, M., Arbeláez, P., & Forero-Romero, J. E. 2020, *MNRAS*, 499, 3130
- Gong, Y., Liu, X., Cao, Y., et al. 2019, *ApJ*, 883, 203
- González, R. E., Munoz, R. P., & Hernández, C. A. 2018, *A&C*, 25, 103
- Gunn, J. E., Siegmund, W. A., Mannery, E. J., et al. 2006, *AJ*, 131, 2332
- Guo, H., Yang, X., Wang, N., & Gao, X. 2021, *PatRe*, 112, 107787
- Hansen, D. L., Mendoza, I., Liu, R., et al. 2022, in *Proc. of the 39th Conf. Machine Learning (ICML 2022)* (Baltimore, MD: Machine Learning for Astrophysics), 27
- Harris, C. R., Millman, K. J., Van Der Walt, S. J., et al. 2020, *Natur*, 585, 357
- Hausen, R., & Robertson, B. E. 2020, *ApJS*, 248, 20
- He, K., Zhang, X., Ren, S., & Sun, J. 2016, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (New York: IEEE)
- Hora, S. C. 1996, *Reliab. Eng. Syst. Saf.*, 54, 217
- Hortúa, H. J., Volpi, R., Marinelli, D., & Malagò, L. 2020, *PhRvD*, 102, 103509
- Hu, L., Wang, L., Chen, X., & Yang, J. 2022, *ApJ*, 936, 157
- Hunter, J. D. 2007, *CSE*, 9, 90
- Ivezić, Z., Kahn, S. M., Tyson, J. A., et al. 2019, *ApJ*, 873, 111
- Jia, P., Liu, Q., & Sun, Y. 2020, *AJ*, 159, 212
- Jia, P., Liu, W., Liu, Y., & Pan, H. 2023a, *ApJS*, 264, 43
- Jia, P., Zheng, Y., Wang, M., & Yang, Z. 2023b, *A&C*, 42, 100687
- Jia, P., Sun, R., Li, N., et al. 2022, *AJ*, 165, 26
- Jia, P., Zhao, Y., Xue, G., & Cai, D. 2019, *AJ*, 157, 250
- Kessler, R., Marriner, J., Childress, M., et al. 2015, *AJ*, 150, 172
- Khrantsov, V., Dobrycheva, D., Vasilenko, M. Y., & Akhmetov, V. 2019, *OAP*, 32, 21
- Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
- Kingma, D. P., & Welling, M. 2013, arXiv:1312.6114
- Kiureghian, A. D., & Ditlevsen, O. 2009, *Structural Safety*, 31, 105
- Krishnan, R., Esposito, P., & Subedar, M. 2022, *Bayesian-Torch: Bayesian Neural Network Layers for Uncertainty Estimation, v0.2.0*, Zenodo, doi:10.5281/zenodo.5908307
- Kuleshov, V., Fenner, N., & Ermon, S. 2018, in *Proc. of the 35th Int. Conf. on Machine Learning*, ed. J. Dy & A. Krause (Stockholm: PMLR), 2796, <https://proceedings.mlr.press/v80/kuleshov18a.html>
- Lang, D., Hogg, D. W., Mierle, K., Blanton, M., & Roweis, S. 2010, *AJ*, 139, 1782
- Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollar, P. 2017, in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)* (New York: IEEE)
- Liu, J., Soria, R., Wu, X.-F., Wu, H., & Shang, Z. 2021, *AnABC*, 93, 20200628
- Liu, R., McAuliffe, J. D., & Regier, J. 2021a, arXiv:2102.02409
- Liu, Z., Lin, Y., Cao, Y., et al. 2021b, in *Proc. of the IEEE/CVF Int. Conf. on Computer Vision* (New York: IEEE), 10012
- Lundh, F. & Contributors 1995-2021, *Pillow Documentation*, <https://pillow.readthedocs.io/en/stable/>
- Lupton, R. H., Ivezić, Z., Gunn, J. E., et al. 2005, *Technical Report*, Princeton Univ.
- Makhlof, K., Turpin, D., Corre, D., et al. 2022, *A&A*, 664, A81
- McKinney, W., et al. 2011, *Python for High Performance and Scientific Computing*, 14, 1
- Mong, Y.-L., Ackley, K., Galloway, D., et al. 2020, *MNRAS*, 499, 6009
- Newell, A., Yang, K., & Deng, J. 2016, in *Computer Vision—ECCV 2016*, ed. B. Leibe et al. (Cham: Springer International Publishing), 483
- Padmanabhan, N., Schlegel, D. J., Finkbeiner, D. P., et al. 2008, *ApJ*, 674, 1217
- Paszke, A., Gross, S., Massa, F., et al. 2019, *Advances in Neural Information Processing Systems*, Vol. 32 (Cambridge, MA: MIT Press)
- Pier, J. R., Munn, J. A., Hindsley, R. B., et al. 2003, *AJ*, 125, 1559

- Ren, S., He, K., Girshick, R., & Sun, J. 2015, *Advances in Neural Information Processing Systems*, Vol. 28 (Cambridge, MA: MIT Press)
- Robitaille, T. P., Tollerud, E. J., Greenfield, P., et al. 2013, *A&A*, **558**, A33
- Rousseuw, P. J., & Croux, C. 1993, *JASA*, **88**, 1273
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1986, *Natur*, **323**, 533
- Sánchez, B., Lares, M., Beroiz, M., et al. 2019, *A&C*, **28**, 100284
- Smith, J. A., Tucker, D. L., Kent, S., et al. 2002, *AJ*, **123**, 2121
- Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. 2014, arXiv:1412.6806
- Stoughton, C., Lupton, R. H., Bernardi, M., et al. 2002, *AJ*, **123**, 485
- Tran, K., Neiswanger, W., Yoon, J., et al. 2020, *MLS&T*, **1**, 025006
- Turpin, D., Ganet, M., Antier, S., et al. 2020, *MNRAS*, **497**, 2641
- Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, *NatMe*, **17**, 261
- Wen, Y., Vicol, P., Ba, J., Tran, D., & Grosse, R. 2018, arXiv:1803.04386
- Wright, D., Smartt, S., Smith, K., et al. 2015, *MNRAS*, **449**, 451
- York, D. G., Adelman, J., Anderson, J. E., Jr., et al. 2000, *AJ*, **120**, 1579
- Yu, P.-p., Sun, R.-y., Yu, S.-x., et al. 2022, *AdSpR*, **70**, 3311
- Zacharias, N., Urban, S. E., Zacharias, M. I., et al. 2000, *AJ*, **120**, 2131
- Zackay, B., Ofek, E. O., & Gal-Yam, A. 2016, *ApJ*, **830**, 27
- Zhou, X., Wang, D., & Krähenbühl, P. 2019, arXiv:1904.07850