# Detection is truncation: studying source populations with truncated marginal neural ratio estimation

**Noemi Anau Montel**
GRAPPA Institute
University of Amsterdam, The Netherlands
n.anaumontel@uva.nl

**Christoph Weniger**
GRAPPA Institute
University of Amsterdam, The Netherlands
c.weniger@uva.nl

## Abstract

Statistical inference of population parameters of astrophysical sources is challenging. It requires accounting for selection effects, which stem from the artificial separation between bright detected and dim undetected sources that is introduced by the analysis pipeline itself. We show that these effects can be modeled self-consistently in the context of sequential simulation-based inference. Our approach couples source detection and catalog-based inference in a principled framework that derives from the truncated marginal neural ratio estimation (TMNRE) algorithm. It relies on the realization that detection can be interpreted as prior truncation. We outline the algorithm, and show first promising results.

## 1 Introduction

Point sources detection is crucial for astronomical surveys, and is the cornerstone for the compilation of source catalogues. Those source catalogues are then typically the basis for the inference of physical parameters that describe the sources at the population level. Upcoming astronomical facilities, such as the Square Kilometer Array (SKA) [1] and the Cherenkov Telescope Array (CTA) [2] will deliver large and complex datasets. In order to leverage their full potential, it is urgent to develop robust and automated source detection and source population parameters inference algorithms.

Recent developments in deep learning and more generally automatic differentiation frameworks [3] are increasingly used for tackling difficult astronomical data analysis challenges. The capability of deep learning techniques of point sources detection and population characterization has been demonstrated across different wavelengths surveys, e.g. in $\gamma$-ray data [4–7], radio data [8–11] and cosmic microwave background data [12]. In particular simulation-based machine learning approaches can be highly flexible, allowing to tailor developed pipelines to specific telescopes and science cases. A range of simulation-based inference (SBI) algorithms have been proposed in the literature (see Cranmer et al. [13] for a review). An appealing feature is that they generally allow to directly estimate marginal posteriors for parameters of interest [14]. Furthermore, *sequential* SBI approaches [15–17] have been shown to be particularly simulation efficient. Among those, truncated marginal neural ratio estimation (TMNRE) [14, 18] is a sequential SBI approach based on neural ratio estimation (NRE) [19], which particularly well composes with marginalization.

Here, we present a strategy for how to use TMNRE [1] to simultaneously perform source detection and population-level parameters inference. This enables to self-consistently combine information from both detected and sub-threshold sources, without being affected by detection biases. The key idea is to recast the traditional concept of *source detection* in terms of *prior truncation*. This will allow us to *distill* information of bright sources directly into the simulation model. Our proposed method is highly interpretable since it resembles components of traditional survey analysis workflows.

[1] We use *swyft* TMNRE implementation that can be found at `https://github.com/undark-lab/swyft`.

## 2 Methodology

**Background** TMNRE (and NRE in general) performs posterior estimation by solving a binary classification problem. Given a model $p(x, z) = p(x|z)p(z)$, where $x$ is data, and $z$ a set of parameters of interest, one trains a network to distinguish joined samples $x, z \sim p(x, z)$ from marginal samples $x, z \sim p(x)p(z)$. The networks learns to estimate the likelihood-to-evidence ratio $r(z; x) = p(x|z)/p(x)$ [2], which we can use to obtain weighted samples from the posterior $p(z|x) = r(z; x)p(z)$. In order to improve the network's learning and maximise the simulator efficiency, TMNRE concentrates in stages the regions in parameter space from which training examples are drawn based on a target observation. Hence, this truncation scheme restricts the prior distribution's support without modifying its shape, as opposed to other sequential methods that employ a posterior estimate as proposal distribution for generating simulations for the next round [15].

**Simulation model** We consider here a simple Bayesian hierarchical source model,

$$p(\boldsymbol{x}, \vec{\boldsymbol{s}}, \boldsymbol{\vartheta}) = p(\boldsymbol{x}|\vec{\boldsymbol{s}})p(\boldsymbol{\vartheta}) \prod_{i=1}^{N} p(\boldsymbol{s}_i|\boldsymbol{\vartheta}) \,, \tag{1}$$

where $\boldsymbol{x}$ is the observed sky map, $\boldsymbol{s}_i \equiv (F_i, \Omega_i)$ denotes the flux $F_i$ and position $\Omega_i$ of point source $i$, and $\boldsymbol{\vartheta} \equiv \{N, \Sigma, h\}$ collects source population parameters, namely the number of sources $N$, and the parameters $\Sigma$ and $h$ that control the flux and spatial distributions respectively. Finally, we add instrumental effects to simulated maps. We give more details on the model in appendix A. We show examples of our simulated maps in the top row of the right panel of fig. 2.

**Source detection** For source detection we consider the following likelihood-to-evidence ratio

$$r_1(\Omega, F_{th}; \boldsymbol{x}) \equiv \frac{p(\mathbb{I}_{\boldsymbol{x}}(F \geq F_{th}) = 1, \Omega|\boldsymbol{x})}{p(\mathbb{I}_{\boldsymbol{x}}(F \geq F_{th}) = 1, \Omega)} \,. \tag{2}$$

Here, the denominator corresponds to the prior probability of having a source at position $\Omega$ with a flux $F \geq F_{th}$ that exceeds some threshold flux $F_{th}$. The numerator is the corresponding posterior. We model the source detection ratio estimator in eq. (2) as an image-to-image neural network that solves a binary classification problem in each image pixel. For simulated data, we call a simulated source $\boldsymbol{s}_i$ 'detected' when there is a corresponding compact region as function of $\Omega$ where the detection significance is above threshold, $r_1(\Omega, F_{th}; \boldsymbol{x}) > 5$. This effectively leads to a split between sources that are clearly identifiable and 'sub-threshold' sources that are difficult to detect a individual instances. Below, we assign the detection label $d_i = 1$ ($d_i = 0$) to detected (undetected) sources.

In order to characterize the split between detected and sub-threshold sources, we introduce a source-sensitivity function, $S(F, \Omega)$, which provides the probability that a source with flux $F$ and at position $\Omega$ would be detected by the ratio estimator in eq. (2). This function can be estimated by training the ratio estimator

$$r_2(d; F, \Omega, \boldsymbol{x}) \equiv \frac{p(d|F, \Omega, \boldsymbol{x})}{p(d)} \tag{3}$$

which is marginalised over all other sources and source parameters. By omitting the dependence on the map, $\boldsymbol{x}$, which is then effectively marginalized, the ratio estimator can then simply be modeled as a $\mathbb{R}^3 \to \mathbb{R}$ multi-layer perceptron. The source-sensitivity function can then be estimated as

$$S(F, \Omega) = \sigma\left(\log\left(\frac{p(d = 1|F, \Omega)}{p(d = 0|F, \Omega)}\right)\right) , \tag{4}$$

where we have introduced the sigmoid function $\sigma(y) \equiv 1/(1 + e^{-y})$.

We can make the concept of source detection part of our model as follows. In a random realization, each source $i$ will be either detected, $d_i = 1$ (with probability $S(F_i, \Omega_i)$), or not detected, $d_i = 0$ (with probability $1 - S(F_i, \Omega_i)$). To keep notation simple, we omit $d_i$ and instead group detected

[2] In the next section we will use the notation $r(a; b|c) = \frac{p(a,b|c)}{p(a|c)p(b|c)}$, where with '$|$' we refer to conditioning on specific variables for all factors in the ratio definition. If necessary, multiple variables are comma separated, for example $r(a, b; c|d) = \frac{p(a,b,c|d)}{p(a,b|d)p(c|d)}$. Training conditional ratios is a straight-forward extension of NRE.

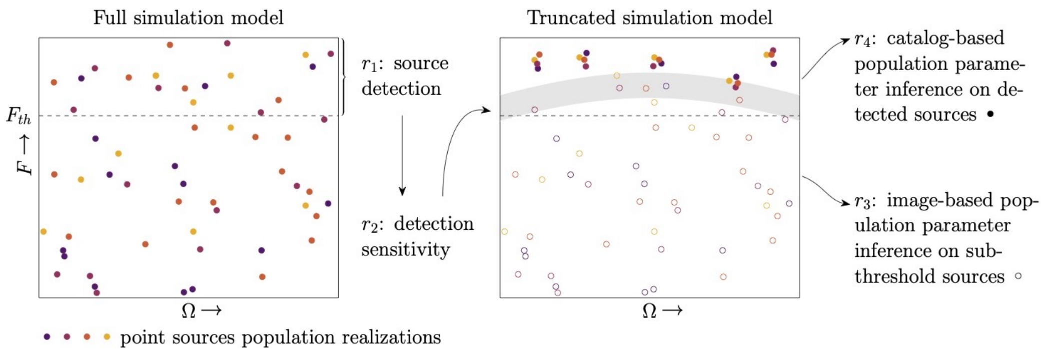
Full simulation model
Truncated simulation model
$r_1$: source detection
$r_2$: detection sensitivity
$r_4$: catalog-based population parameter inference on detected sources •
$r_3$: image-based population parameter inference on sub-threshold sources ○
$F_{th}$
$F \rightarrow$
$\Omega \rightarrow$
point sources population realizations

Figure 1: Illustration of our inference framework (see section 2 for details). *Left panel*: A source detection network $r_1$ and the corresponding sensitivity network $r_2$ are trained based on the full simulation model. *Right panel*: Bright sources are constrained in the truncated simulation model, while sub-threshold sources vary freely. Two inference networks are trained to capture information from sub-threshold sources ($r_3$) and detected sources ($r_4$).

and non-detected (sub-treshold) sources together in vectors with corresponding subscripts, and write $\vec{\boldsymbol{s}} \equiv (\vec{\boldsymbol{s}}_{det}, \vec{\boldsymbol{s}}_{sub})$. Taking into consideration this split, our model becomes

$$p(\boldsymbol{x}, \vec{\boldsymbol{s}}_{det}, \vec{\boldsymbol{s}}_{sub}, \boldsymbol{\vartheta}) = p(\boldsymbol{x}|\vec{\boldsymbol{s}}_{det}, \vec{\boldsymbol{s}}_{sub}) p(\vec{\boldsymbol{s}}_{det}|\boldsymbol{\vartheta}) p(\vec{\boldsymbol{s}}_{sub}|\boldsymbol{\vartheta}) p(\boldsymbol{\vartheta}) \ . \tag{5}$$

Importantly, although $p(\vec{\boldsymbol{s}}_{det/sub}|\boldsymbol{\vartheta})$ depends on the sensitivity function $S(F_i, \Omega_i)$, the distribution of all sources $p(\vec{\boldsymbol{s}}|\boldsymbol{\vartheta})$ is the same as in eq. (1).

**Detection as truncation** Truncating source priors in eq. (1) is difficult due to the the label switching problem ($\boldsymbol{x}$ is invariant under relabeling sources). However, once specific sources are detected, they can be labeled and ordered arbitrarily. Let us assume that $N_{det}$ sources were detected by the ratio estimator in eq. (2) in the regions $\mathcal{R}_i$ with $i = 1, \dots, N_{det}$ for a given observation of reference $\boldsymbol{x}_o$. Our ansatz for the indicator function, which selects a specific prior region, is

$$\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = \prod_{i=1}^{N_{det}} \mathbb{I}_{\boldsymbol{x}_o}(\Omega_i \in \mathcal{R}_i) \mathbb{I}_{\boldsymbol{x}_o}(F_i \geq F_{th}) \ . \tag{6}$$

Our truncation strategy is now to focus on the parameter space where $\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1$ for our data of interest $\boldsymbol{x}_o$. We can then write our truncated model as

$$p(\boldsymbol{x}, \vec{\boldsymbol{s}}_{det}, \vec{\boldsymbol{s}}_{sub}, \boldsymbol{\vartheta}, \mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det})) = p(\boldsymbol{x}|\vec{\boldsymbol{s}}_{det}, \vec{\boldsymbol{s}}_{sub}) p(\vec{\boldsymbol{s}}_{det}|\boldsymbol{\vartheta}, \mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det})) p(\vec{\boldsymbol{s}}_{sub}|\boldsymbol{\vartheta}) p(\boldsymbol{\vartheta}) p(\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det})|\boldsymbol{\vartheta}). \tag{7}$$

**Population parameters inference with truncation** Given some observation $\boldsymbol{x}_o$, we want to estimate the posterior $p(\boldsymbol{\vartheta}|\boldsymbol{x}_o)$. Since the truncation affects parameters $\vec{\boldsymbol{s}}_{det}$ whose prior depends on population parameters $\boldsymbol{\vartheta}$, the truncation volume is not a constant factor, and the procedure requires extra care. We can estimate the posterior $p(\boldsymbol{\vartheta}|\boldsymbol{x})$ by considering the ratio

$$r(\boldsymbol{\vartheta}; \boldsymbol{x}) = \frac{p(\boldsymbol{\vartheta}|\boldsymbol{x})}{p(\boldsymbol{\vartheta})} \simeq \frac{p(\boldsymbol{\vartheta}|\boldsymbol{x}, \mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)}{p(\boldsymbol{\vartheta})} = \underbrace{\frac{p(\boldsymbol{\vartheta}|\boldsymbol{x}, \mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)}{p(\boldsymbol{\vartheta}|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)}}_{\equiv r_3(\boldsymbol{\vartheta}; \boldsymbol{x}|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det})=1)} \underbrace{\frac{p(\boldsymbol{\vartheta}|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)}{p(\boldsymbol{\vartheta})}}_{\equiv r(\boldsymbol{\vartheta}; \mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det})=1)} \ . \tag{8}$$

The second step in eq. (8) corresponds to the truncation approximation. It is exact (in the sense of leaving $p(\boldsymbol{\vartheta}|\boldsymbol{x}_o)$ unaffected) in the limit where $p(\boldsymbol{x}_o|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 0) \to 0$. Training that ratio estimator directly with TMNRE would be challenging since $\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1$ has very small support in the training data. Instead, we split the ratio into two computationally feasible ratios (this is in spirit similar to the telescoping ratio estimation approach presented in Rhodes et al. [20]).

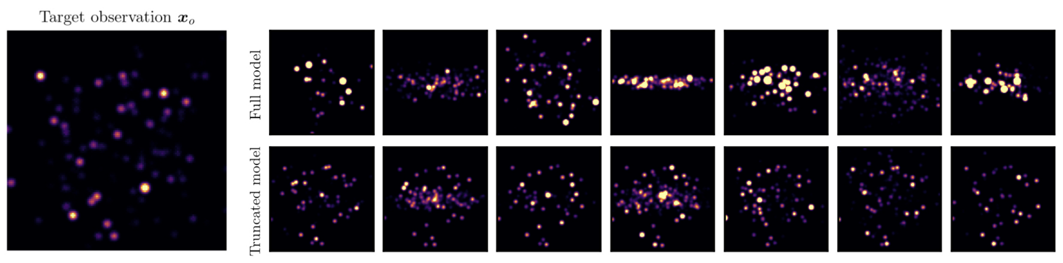

Figure 2: *Left:* Observation of reference $\boldsymbol{x}_o$. *Right:* In the first row we show samples from our full simulation model. In the second row we show targeted samples from the truncated model. Targeted data are visually more close to $\boldsymbol{x}_o$, the main dissimilarities are due to different sub-threshold sources and instrumental effects realizations.

The first ratio, $r_3(\boldsymbol{\vartheta}; \boldsymbol{x}|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)$, can be estimated by training a peak-count network [21–23] on targeted data, that is truncated to $\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1$. The second ratio can be estimated as

$$\begin{aligned} r(\boldsymbol{\vartheta}; \mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1) &= \frac{p(\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1|\boldsymbol{\vartheta})}{p(\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)} \\ &= \int d\vec{\boldsymbol{s}}_{det} \frac{p(\vec{\boldsymbol{s}}_{det}|\boldsymbol{\vartheta})}{p(\vec{\boldsymbol{s}}_{det})} \frac{p(\vec{\boldsymbol{s}}_{det})\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det})}{p(\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)} = \int d\vec{\boldsymbol{s}}_{det} \underbrace{\frac{p(\vec{\boldsymbol{s}}_{det}|\boldsymbol{\vartheta})}{p(\vec{\boldsymbol{s}}_{det})}}_{\equiv r_4(\vec{\boldsymbol{s}}_{det};\boldsymbol{\vartheta})} p(\vec{\boldsymbol{s}}_{det}|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1) \ , \end{aligned} \quad (9)$$

so it can be estimated by training a network on detected sources lists on un-truncated training data. In practice, we generate weighted samples from the full posterior $p(\boldsymbol{\vartheta}|\boldsymbol{x})$ by sampling $\boldsymbol{\vartheta}, \vec{\boldsymbol{s}}_{det} \sim p(\boldsymbol{\vartheta})p(\vec{\boldsymbol{s}}_{det}|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)$ with weights $w = r_3(\boldsymbol{\vartheta}; \boldsymbol{x}|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1) \cdot r(\boldsymbol{\vartheta}; \mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)$.

In the process, we trained four ratio estimation networks that are directly connected with traditional source analysis pipeline components: $r_1(\Omega, F_{th}; \boldsymbol{x})$ performs source detection; $r_2(d; F, \Omega, \boldsymbol{x})$ is the source sensitivity function; $r_3(\boldsymbol{\vartheta}; \boldsymbol{x}|\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1)$ constraints $\boldsymbol{\vartheta}$ based on sub-threshold sources (because detected sources are assumed to be fixed in the parameter space where $\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1$); and $r_4(\vec{\boldsymbol{s}}_{det}; \boldsymbol{\vartheta})$ constraints $\boldsymbol{\vartheta}$ using the detected sources catalog. We show a schematic overview of the inference framework used in this work in fig. 1.

## 3 Results

We apply the proposed methodology to the target observation $\boldsymbol{x}_o$ shown in fig. 2. We first train the source detection ratio estimator in eq. (2) on data simulated from the full model shown in eq. (1), and then apply it to $\boldsymbol{x}_o$ to obtain a detection map. From the detection map we derive a catalog of detected sources $\vec{\boldsymbol{s}}_{det}$, and define a truncated parameter space of interest, where $\mathbb{I}_{\boldsymbol{x}_o}(\vec{\boldsymbol{s}}_{det}) = 1$. In order to make source detection part of our model, we train the sensitivity ratio estimator in eq. (3) to estimate the sensitivity function $S(F, \Omega)$. We then generate targeted training data from our truncated simulation model in eq. (7). We show samples from the full model and the truncated one in fig. 2.

Finally, we train two inference networks to capture information regarding population parameters from sub-threshold and detected sources, as explained in section 2. We show the constraints on population parameters $\boldsymbol{\vartheta}$ from sub-threshold sources, detected ones, and their combination in fig. 3. The different posteriors are consistent with each other, indicating that the proposed inference framework automatically accounts for detection biases. We see that different constraints are dominated by different neural networks, e.g. the number $N$ of point source is better constrained by sub-threshold ones, whereas the spatial distribution parameter $h$ by detected ones. The weaker constraint on the flux parameter $\Sigma$ inferred from detected sources is due to the fact that in eq. (9) we average over different detected sources realisations, always re-sampling the parameter $\Sigma$ (see appendix A for the hierarchical model details).

The four neural networks were trained on a NVIDIA GeForce RTX 3080 Ti GPU, the total computational time cost to obtain the results shown in fig. 3 is $\sim 2$ hours. We intend on describing in details the networks used for each task and choice of hyperparameters in future work.

## 4 Conclusions

We have introduced a novel method to self-consistently perform point sources detection and source population parameters inference using TMNRE. With this approach, we can exploit information of detected as well as sub-threshold sources for population-level parameter inference. Detection biases are automatically accounted for in our approach. Exemplary results of our approach are shown in fig. 3, where we show inference results on source population parameters from both detected point sources and sub-threshold sources separately, as well as their combination. Since the proposed method is essentially a specific implementation of TMNRE, we expect that it inherits its positive properties in terms of simulation-efficiency and scalability [18]. A possible shortcoming of this approach is that multiple neural networks need to be trained self-consistently, which on the other hand have a clear interpretation in terms of traditional analysis pipeline components. A potential application beyond those directly intended is to detectable and sub-threshold substructures in strong gravitational lenses.

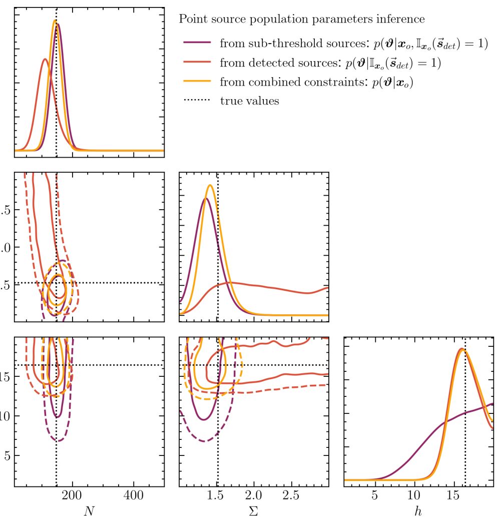

Figure 3: Marginal posteriors inferred by TMNRE on target observation $\boldsymbol{x}_o$ for population parameters $\boldsymbol{\vartheta}$. We show constraints from sub-threshold sources (violet), detected sources (orange), and combined results (yellow). In the 2D marginal posterior we indicate the 68% and 95% credible regions with solid and dashed lines respectively. For more details see section 3.

**Broader Impact** This work is focusing on the analysis of astronomical surveys that contain point sources population via TMNRE. Variants of the presented approach could find application in other areas of the physical sciences. We do not expect any negative societal impact of the presented methods. However, we recommend the usual caution in inferring scientific conclusions based on a complex methodology.

## Acknowledgments and Disclosure of Funding

This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 864035 – Undark).

We acknowledge the use of the `python` [24] modules, `matplotlib` [25], `numpy` [26], `scipy` [27], `PyTorch` [28], `tqdm` [29], and `jupyter` [30].

## References

[1] A. Weltman et al. Fundamental physics with the Square Kilometre Array. *Publ. Astron. Soc. Austral.*, 37:e002, 2020. doi: 10.1017/pasa.2019.42.

[2] B. S. Acharya et al. *Science with the Cherenkov Telescope Array*. WSP, 11 2018. ISBN 978-981-327-008-4. doi: 10.1142/10986.

[3] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. 2015. doi: 10.48550/ARXIV.1502.05767. URL `https://arxiv.org/abs/1502.05767`.

[4] Boris Panes, Christopher Eckner, Luc Hendriks, Sacha Caron, Klaas Dijkstra, Guðlaugur Jóhannesson, Roberto Ruiz de Austri, and Gabrijela Zaharijas. Identification of point sources in gamma rays using u-shaped convolutional neural networks and a data challenge. *Astronomy*

*& Astrophysics*, 656:A62, nov 2021. doi: 10.1051/0004-6361/202141193. URL `https://doi.org/10.1051%2F0004-6361%2F202141193`.

[5] Sascha Caron, Germá n A. Gómez-Vargas, Luc Hendriks, and Roberto Ruiz de Austri. Analyzing $\gamma$ rays of the galactic center with deep learning. *Journal of Cosmology and Astroparticle Physics*, 2018(05):058–058, may 2018. doi: 10.1088/1475-7516/2018/05/058. URL `https://doi.org/10.1088%2F1475-7516%2F2018%2F05%2F058`.

[6] Florian List, Nicholas L. Rodd, and Geraint F. Lewis. Extracting the galactic center excess' source-count distribution with neural nets. *Physical Review D*, 104(12), dec 2021. doi: 10.1103/physrevd.104.123022. URL `https://doi.org/10.1103%2Fphysrevd.104.123022`.

[7] Siddharth Mishra-Sharma and Kyle Cranmer. Neural simulation-based inference approach for characterizing the galactic center $\gamma$-ray excess. *Physical Review D*, 105(6), mar 2022. doi: 10.1103/physrevd.105.063017. URL `https://doi.org/10.1103%2Fphysrevd.105.063017`.

[8] A. Vafaei Sadr, Etienne E. Vos, Bruce A. Bassett, Zafiirah Hosenie, N. Oozeer, and Michelle Lochner. DeepSource: Point Source Detection using Deep Learning. *Mon. Not. Roy. Astron. Soc.*, 484(2):2793–2806, 2019. doi: 10.1093/mnras/stz131.

[9] S Rezaei, J P McKean, M Biehl, and A Javadpour. DECORAS: detection and characterization of radio-astronomical sources using deep learning. *Monthly Notices of the Royal Astronomical Society*, 510(4):5891–5907, dec 2021. doi: 10.1093/mnras/stab3519. URL `https://doi.org/10.1093%2Fmnras%2Fstab3519`.

[10] V. Lukic, F. De Gasperin, and M. Brüggen. Convosource: Radio-astronomical source-finding with convolutional neural networks, 2019. URL `https://arxiv.org/abs/1910.03631`.

[11] Duncan Tilley, Christopher W. Cleghorn, Kshitij Thorat, and Roger Deane. Point proposal network: Accelerating point source detection through deep learning. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, dec 2021. doi: 10.1109/ssci50451.2021.9660085. URL `https://doi.org/10.1109%2Fssci50451.2021.9660085`.

[12] L. Bonavera, S. L. Suarez Gomez, J. González-Nuevo, M. M. Cueli, J. D. Santos, M. L. Sanchez, R. Muñiz, and F. J. de Cos. Point source detection with fully convolutional networks. performance in realistic microwave sky simulations. *A&A*, 648:A50, April 2021. doi: 10.1051/0004-6361/201937171.

[13] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, may 2020. doi: 10.1073/pnas.1912789117. URL `https://doi.org/10.1073%2Fpnas.1912789117`.

[14] Benjamin Kurt Miller, Alex Cole, Gilles Louppe, and Christoph Weniger. Simulation-efficient marginal posterior estimation with swyft: stop wasting your precious time. *arXiv preprint arXiv:2011.13951*, 2020.

[15] Conor Durkan, George Papamakarios, and Iain Murray. Sequential neural methods for likelihood-free inference, 2018. URL `https://arxiv.org/abs/1811.08723`.

[16] George Papamakarios and Iain Murray. Fast $\epsilon$-free inference of simulation models with bayesian conditional density estimation, 2016. URL `https://arxiv.org/abs/1605.06376`.

[17] George Papamakarios, David C. Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows, 2018. URL `https://arxiv.org/abs/1805.07226`.

[18] Benjamin K Miller, Alex Cole, Patrick Forré, Gilles Louppe, and Christoph Weniger. Truncated marginal neural ratio estimation. *Advances in Neural Information Processing Systems*, 34: 129–143, 2021.

[19] Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with Amortized Approximate Ratio Estimators. 3 2019.

[20] Benjamin Rhodes, Kai Xu, and Michael U. Gutmann. Telescoping density-ratio estimation, 2020. URL https://arxiv.org/abs/2006.12204.

[21] Hisham Cholakkal, Guolei Sun, Fahad Shahbaz Khan, and Ling Shao. Object counting and instance segmentation with image-level supervision, 2019. URL https://arxiv.org/abs/1903.02494.

[22] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything, 2021. URL https://arxiv.org/abs/2104.08391.

[23] Ersin Kilic and Serkan Ozturk. An accurate car counting in aerial images based on convolutional neural networks. *Journal of Ambient Intelligence and Humanized Computing*, jul 2021. doi: 10.1007/s12652-021-03377-5. URL https://doi.org/10.1007%2Fs12652-021-03377-5.

[24] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[25] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9 (3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[26] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, Sep 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. URL http://dx.doi.org/10.1038/s41586-020-2649-2.

[27] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[29] Casper da Costa-Luis, Stephen Karl Larroque, Kyle Altendorf, Hadrien Mary, richardsheridan, Mikhail Korobov, Noam Yorav-Raphael, Ivan Ivanov, Marcel Bargull, Nishant Rodrigues, Guangshuo CHEN, Antony Lee, Charles Newey, James, Joshua Coales, Martin Zugnoni, Matthew D. Pagel, mjstevens777, Mikhail Dektyarev, Alex Rothberg, Alexander, Daniel Panteleit, Fabian Dill, FichteFoll, Gregor Sturm, HeoHeo, Hugo van Kemenade, Jack McCracken, MapleCCC, and Max Nordlund. tqdm: A fast, Extensible Progress Bar for Python and CLI, September 2021. URL https://doi.org/10.5281/zenodo.5517697.

[30] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

Table 1: Point source simulation model parameters and priors.

| Parameter | | Prior |
|---|---|---|
| Population parameters | | |
| number of point sources | $N$ | $\mathcal{U}(10, 500)$ |
| flux distribution parameter | $\Sigma$ | $\mathcal{U}(1, 3)$ |
| spatial distribution parameter | $h$ | $\mathcal{U}(1, 20)$ |
| Point source parameters | | |
| flux | $F$ | $\log \mathcal{N}(1, \Sigma)$ |
| position | $\Omega \equiv (l, b)$ | $(\mathcal{N}(0, 20), \mathcal{N}(0, h))$ |

## A Appendix: Simulation model

We describe in more detail the Bayesian hierarchical point source model adopted in this work. To generate an observation, first, we sample point sources population parameters $\boldsymbol{\vartheta} \equiv \{N, \Sigma, h\}$ from their priors, given in table 1. Then, for each point source, we draw its flux $F$ and position on the map $\Omega \equiv (l, b)$ from the priors given in table 1. We then generate a $128 \times 128$ pixels map. To model instrumental effects, we add a point-spread function (PSF) with Gaussian kernel standard deviation $\epsilon = 1.5$ and Poisson noise, obtaining the final simulated map $\boldsymbol{x}$.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] As stated in the abstract and contribution, the main scope of this work is to propose the methodology described in section 2, whose exemplary results we show in section 3
   (b) Did you describe the limitations of your work? [Yes] We have presented work-in-progress results and briefly mentioned possible shortcomings of this method in section 4.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] The proposed methodology applies to astronomical data, we do not anticipate any negative societal impacts from it. However, we have recommended caution when using a complex analysis machinery in section 4.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Our experiments were run using publicly available code, while our simple simulation model is described in details in appendix A. We plan on describing the networks used for each task in future works.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [No]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We only use open-source code and data simulated from our model.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] The data is generated by a physical models which abstractly relate parameters to astronomical data.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]