Neural network emulator of the Advanced LIGO and Advanced Virgo selection function

Thomas A. Callister,[1] Reed Essick,[2,3,4] and Daniel E. Holz[1,5,6,7]

[1]*Kavli Institute for Cosmological Physics, The University of Chicago, 5640 South Ellis Avenue, Chicago, Illinois 60615, USA*
[2]*Canadian Institute for Theoretical Astrophysics, 60 St. George Street, Toronto, Ontario M5S 3H8, Canada*
[3]*Department of Physics, University of Toronto, 60 St. George Street, Toronto, Ontario M5S 1A7, Canada*
[4]*David A. Dunlap Department of Astronomy, University of Toronto, 50 St. George Street, Toronto, Ontario M5S 3H4, Canada*
[5]*Department of Physics, The University of Chicago, Chicago, Illinois 60637, USA*
[6]*Department of Astronomy and Astrophysics, The University of Chicago, Chicago, Illinois 60637, USA*
[7]*Enrico Fermi Institute, The University of Chicago, Chicago, Illinois 60637, USA*

(Received 3 September 2024; accepted 1 November 2024; published 27 December 2024)

Characterization of search selection effects comprises a core element of gravitational-wave data analysis. Knowledge of selection effects is needed to predict observational prospects for future surveys and is essential in the statistical inference of astrophysical source populations from observed catalogs of compact binary mergers. Although gravitational-wave selection functions can be directly measured via injection campaigns—the insertion and attempted recovery of simulated signals added to real instrumental data—such efforts are computationally expensive. Moreover, the inability to interpolate *between* discrete injections limits the ability to which we can study narrow or discontinuous features in the astrophysical distribution of compact binary properties. For this reason, there is a growing need for alternative representations of gravitational-wave selection functions that are computationally cheap to evaluate and can be computed across a continuous range of compact binary parameters. In this paper, we describe one such representation. Using pipeline injections performed during Advanced LIGO and Advanced Virgo's third observing run (O3), we train a neural network emulator for $P(\det|\theta)$, the probability that a given compact binary with parameters is successfully detected, averaged over the course of O3. The emulator captures the dependence of $P(\det|\theta)$ on binary masses, spins, distance, sky position, and orbital orientation, and it is valid for compact binaries with component masses between 1 and $100M_{\odot}$. We test the emulator's ability to produce accurate distributions of detectable events, and demonstrate its use in hierarchical inference of the binary black hole population.

DOI: 10.1103/PhysRevD.110.123041

I. BACKGROUND

Like most astronomical experiments, gravitational-wave astronomy suffers from selection biases: the Advanced LIGO [1] and Advanced Virgo [2] instruments most readily detect compact binary mergers that are massive, nearby, and situated in preferred sky positions and orientations [3–6]. Unlike many other experiments, however, these selection biases are almost exactly quantifiable. The gravitational-wave signatures of merging compact binaries are, in most cases, described by vacuum general relativity alone and can therefore be calculated from first principles to a high degree of accuracy. These calculated waveforms can then be injected, either via hardware or software, into real instrumental data and analyzed with search pipelines [7–14], allowing us to accurately replicate the survey and directly determine how often signals are missed or found [15–19].

Such knowledge of the gravitational-wave selection function is essential to any physical or astrophysical interpretation of gravitational-wave data. The selection function is needed to forward model and predict future observations, given models for the compact binary population and the physics governing it [e.g., [20–25]]. In the reverse direction, the selection function is a critical ingredient in inference: the reverse-engineering of intrinsic source populations from incomplete and noisy catalogs of detected gravitational-wave signals [18,19,26–28].

In principle, injection campaigns allow the selection function to be calculated to high precision (though still subject to systematic uncertainties like imperfect detector calibration or imperfect gravitational waveform models). In practice, however, we rapidly run into problems of dimensionality and scale. Because compact binary mergers are described by at least 15 parameters (the components' masses and spins, binary position and orientation, etc.), it is impractical to directly compute detection probabilities for every possible combination of binary properties. Additionally, processing mock signals injected into data is computationally expensive and labor intensive, requiring at least as much time and person-power as the actual searches for real gravitational waves. The field therefore relies on several widespread shortcuts to estimate selection functions and detection probabilities. Inference of the compact binary population, for example, usually relies on a large suite of reference injections, whose properties are randomly drawn from an astrophysically plausible distribution of compact binary parameters [29–33]. This fixed injection suite can be importance (re) sampled to target other distributions that are sufficiently "close" to the chosen reference distribution, and various integrals over the space of detectable binary parameters can be approximated as Monte Carlo averages over this discrete set of injections [32,34].

There are difficulties with this paradigm, however. In the forward direction, a fixed set of reference injections cannot be immediately used to quantify the detectability of some new gravitational-wave source not included among the original injections. In the reverse direction, the use of reference injections in population inference may become computationally intractable as gravitational-wave catalogs continue to grow in size. In order for systematic uncertainties in search selection functions to remain subdominant, it has been estimated that the number of reference injections must scale at least linearly with the number of detected gravitational-wave events, although sublinear scaling may be achievable with the use of low-discrepancy sequences [32,35].

Due to the computational requirements of pipeline injections, a source's anticipated signal-to-noise ratio (SNR) is a common semianalytic proxy for detectability. In this case, one calculates the optimal SNR ρ_{opt} of a given source, or, preferably, an "observed" SNR $\hat{\rho}$ that includes the effects of random noise fluctuations, and demands that detections exceed some detection threshold ρ_{thresh} [34,36–39]. However, although large signal-to-noise ratios are a necessary condition for detection, they are not *sufficient*; gravitational-wave detection additionally relies on auxiliary signal consistency checks not captured by simple SNR estimates [e.g., [40]], and instrumental noise more readily mimics some types of signals than others. It is likely that these higher order effects can be semianalytically mimicked by adopting a *variable* SNR threshold that depends on the given source parameters [34], but further development is needed.

In this paper we describe an alternative paradigm for evaluating and correcting for selection effects in gravitational-wave astronomy. Specifically, we construct and demonstrate the use of a neural network that is trained to emulate the Advanced LIGO and Advanced Virgo selection function during the most recent "O3" observing run [33,41,42]. Although trained on a suite of discrete injections analyzed by compact binary search pipelines, the emulator instead learns the latent, continuous function characterizing detection probabilities as a function of compact binary parameters (see Fig. 1). The result is a means of quickly and inexpensively "interpolating" between reference injections. This allows for simple and precise statements regarding the detectability of new gravitational-wave signals and source populations when forward modeling future observations. In the reverse direction, the emulator offers a scalable and computationally feasible route towards population inference with ever-growing gravitational-wave catalogs.

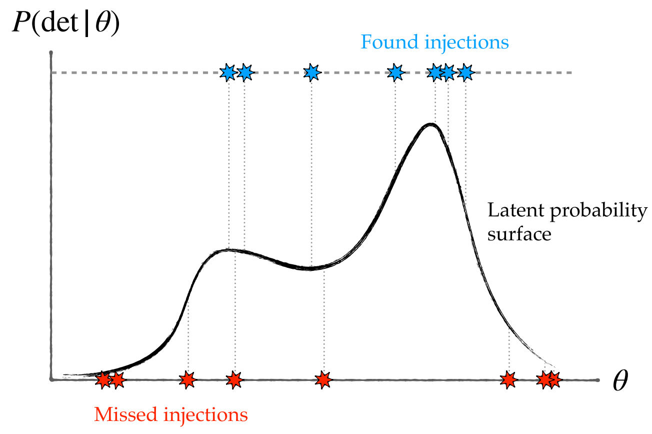

FIG. 1. A cartoon illustrating the latent detection probability surface we seek to model using machine learning, and the discrete realization of missed and found events that serve as our training data. For a given gravitational-wave signal with source parameters θ and a random time of arrival, there exists some probability $P(\det|\theta)$ that it will be successfully detected. Knowledge of $P(\det|\theta)$ is required both to forward model the detectability of different theorized source populations and to infer underlying populations from observed catalogs.

The rest of this paper is organized as follows. In Sec. II, we describe the construction and training of our neural network emulator. In Sec. III, we then demonstrate the trained network's ability to accurately predict distributions of detected events and as well as integrated detection efficiencies. In Sec. IV, we discuss the emulator's use in hierarchical inference. We carry out and compare inference of the binary black hole population using (*i*) standard injections and (*ii*) the trained emulator. We conclude in Sec. VI with a comparison to existing work and a discussion of future applications.

II. TRAINING A DETECTION PROBABILITY EMULATOR

Our goal is to learn the probability $P(\det|\theta)$ that a given gravitational-wave source, with parameters θ, would have been detected if it occurred at a random time during the Advanced LIGO and Advanced Virgo O3 observing run.

The situation is illustrated in Fig. 1. Abstractly, there exists some surface $P(\det|\theta)$ that defines the probability of gravitational-wave detection. We do not have direct access to this surface, however, but instead only indirect information via the locations of successfully detected ("found") and undetected ("missed") mock signals injected into gravitational-wave data with various values of θ [33]. Each such injection formally serves as an individual Bernoulli trial, sampling the underlying $P(\det|\theta)$ surface and returning a single success or a single failure. This process can be repeated at many different times to sample different realizations of instrumental noise and/or periods of terrestrial contamination.

We attempt to learn this latent $P(\det|\theta)$ surface using a simple feed-forward artificial neural network. We adopt a standard multilayer perceptron architecture that takes in a vector of binary parameters θ, parses this input via several densely-connected layers, and concludes with a single output neuron. The output neuron has sigmoid activation function, yielding an estimated detection probability $\hat{P}(\det|\theta) \in \{0, P_{\max}\}$, where $P_{\max} = 0.94$ is the approximate fraction of time that one or both of the LIGO-Hanford and LIGO-Livingston detectors were online during O3 [41]. This threshold reflects the fact that arbitrarily loud signals would still be missed 6% of the time, arriving when neither detector is active.[1] The exact architecture used does not significantly impact the emulator's performance, but we find the best results when adopting four hidden layers with 192 neurons each.

In the following subsections, we describe our training process in more detail, including the training data and loss functions used. Creating and training even a very simple feed-forward neural network, however, requires a multitude of additional design choices, such as the precise balance of training datasets, activation functions, neuron initialization, and learning rate. Details like these are described further in Appendix B.

A. Training Data

We take as training data the set of publicly released software injections performed during the Advanced LIGO and Advanced Virgo O3 observing run [19,33,42]. These comprise simulated gravitational-wave signals added into real O3 data and subsequently passed through several detection pipelines [7–14]. Three such injection sets were prepared, representing the populations of intermediate- and stellar-mass binary black holes, neutron star-black hole binaries, and binary neutron stars; the source-frame chirp masses and luminosity distances of these three datasets are highlighted in the left-hand side of Fig. 2. We consider an injection to have been "detected" if it was assigned a false-alarm rate (FAR) of less than one per year in at least one search pipeline. This matches the selection criterion adopted by the LIGO-Virgo-KAGRA Collaboration in its analysis of the compact binary population following O3 [18,19].

When the O3 injection sets were constructed, only events with a nontrivial chance of being detected (more precisely, events with expected network signal-to-noise ratios exceeding 6) were retained [18,33]. This causes the absence of low-mass injections at large distances in the left-hand side of Fig. 2. Meanwhile, because of the volumetric prior and mass distribution used to generate injections, vanishingly few events are situated at very close distances with large masses. In order to accurately learn $P(\det|\theta)$, we also need training data representing these very weak and very loud signals. We therefore augment the O3 injection sets in two ways. First, we generate several additional injection sets, but this time retain only events that *fail* the network SNR cut (that is, "hopeless" events that are certain to be undetectable). These events, shown in the center column of Fig. 2, are all marked as "missed." Second, we generate sets of injections with expected matched-filtering SNRs of at least 20 in one or both of the Hanford and Livingston detectors. Shown in the right-hand side of Fig. 2, we regard these as "certain" detections, effectively guaranteed to be found provided that one or more detectors are operating at the time of their arrival. We mark ~6% of the certain detections as "missed." As discussed above, this is the percentage of detections that would arrive when neither neither Hanford nor Livingston were operational [41], and are therefore necessarily unobservable. The remainder of the certain detections are labeled as "found." See Appendix B 1 for a further description of these additional injection sets.

B. Defining a loss function

We train our $P(\det|\theta)$ emulator by maximizing the likelihood of having obtained the recorded outcomes (missed or found) of the injections described above.

Let the set $\{\theta_i\}$ describe the parameters of all injections in our training data and $\{\lambda_i\}$ be a set of binary flags recording whether each injection was found ($\lambda_i = 1$) or missed ($\lambda_i = 0$). Given a model $\hat{P}(\det|\theta)$, the likelihood of having obtained this particular realization of missed and found injections is

$$\begin{aligned} p(\{\lambda_i\}|\hat{P}(\det|\theta)) &= \prod_{\text{Found}\,i} \hat{P}(\det|\theta_i) \prod_{\text{Missed}\,j} [1 - \hat{P}(\det|\theta_j)] \\ &= \prod_i [\hat{P}(\det|\theta_i)]^{\lambda_i} [1 - \hat{P}(\det|\theta_i)]^{1-\lambda_i}. \end{aligned} \tag{1}$$

[1]Although this estimate neglects time in which only the Advanced Virgo instrument was online, the Advanced Virgo instrument was significantly less sensitive than the Advanced LIGO instruments in O3, and so the error in this approximation is likely small.

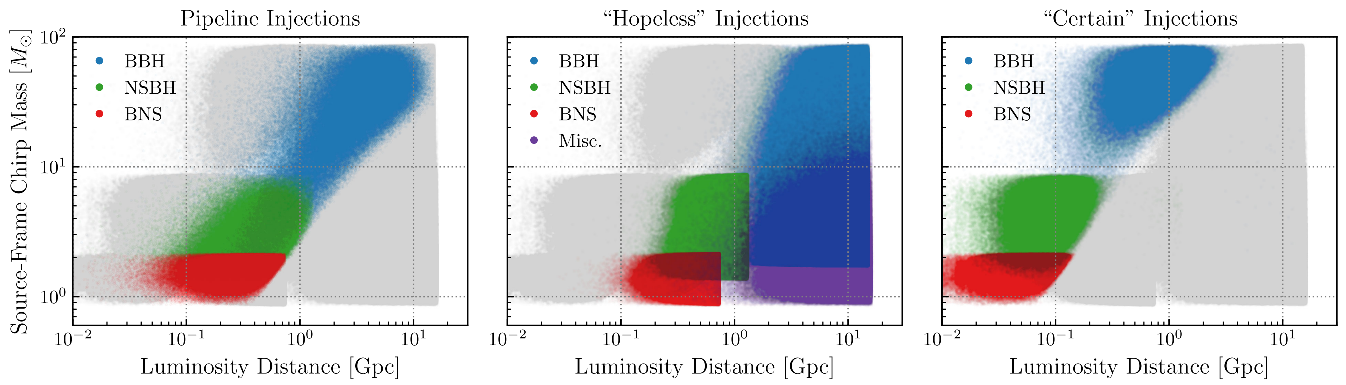

FIG. 2. Summary of the training data used in constructing our emulator for the Advanced LIGO and Advanced Virgo selection function. Shown in the left panel are the source-frame chirp masses and luminosity distances of artificial signals injected into LIGO-Virgo data and processed by compact binary search pipelines to determine which were missed and found. Several such populations were produced and analyzed, corresponding to binary black holes (blue), neutron-star black hole binaries (green), and binary neutron stars (red). We augment these with additional sets of "hopeless" injections, with no chance of successful detection; these are highlighted in the middle panel. We additionally train on sets of "certain" injections, shown in the right panel, which are nearly certain to be identified provided one or more Advanced LIGO detectors are operational at the time of their arrival. Although we display training data in the mass-distance plane here, our emulator is trained on the comprehensive space of binary masses, spins, and extrinsic parameters. Note that the "hopeless," "certain," and pipeline injections do not cleanly separate in the chirp mass-luminosity distance plane; this is due to the influence of these other parameters on compact binary detection.

Rather than maximize this likelihood, we find it useful to instead maximize the posterior $p(\hat{P}(\det|\theta)|\{\lambda\})$, with nontrivial priors on predicted detection probabilities,

$$p(\hat{P}(\det|\theta)|\{\lambda\}) \propto \prod_i e^{-\beta\hat{P}(\det|\theta_i)}[\hat{P}(\det|\theta_i)]^{\lambda_i}[1-\hat{P}(\det|\theta_i)]^{1-\lambda_i}, \tag{2}$$

for some constant β. The parameter β functions to disfavor large detection probabilities, particularly in regions of parameter space with sparse training data. The corresponding loss function is

$$\begin{aligned}\mathcal{L}(\hat{P}(\det|\theta)) &= -\ln p(\hat{P}(\det|\theta)|\{\lambda\}) \\ &= \sum_i [\beta\hat{P}(\det|\theta_i) - \lambda_i \ln \hat{P}(\det|\theta_i) \\ &\quad - (1-\lambda_i)\ln(1-\hat{P}(\det|\theta_i))].\end{aligned} \tag{3}$$

This is the standard binary cross-entropy loss function commonly used in classification, with the additional β-dependent penalization. In the absence of this penalization, our network overpredicts detection probabilities at large distances and low masses where we have few samples; we empirically find that choosing $\beta = 0.35$ alleviates this.

As we will discuss further below, hierarchical inference of the compact binary population does not depend on $P(\det|\theta)$ directly, but instead on the *integral* of $P(\det|\theta)$ over the full parameter space of compact binaries:

$$\xi(\Lambda) \equiv P(\det|\Lambda) = \int d\theta P(\det|\theta)p(\theta|\Lambda). \tag{4}$$

Here, $p(\theta|\Lambda)$ is the probability distribution of binary parameters according to some specific model for the compact binary population, denoted Λ, and $\xi(\Lambda)$ is the fraction of all binaries in this population we expect to successfully detect. This quantity is also called the *detection efficiency*. We find that neural networks trained to optimize pointwise detection probabilities, using Eq. (3), do not necessarily yield good estimates of the integrated detection efficiencies needed for population inference. We therefore augment the loss function to explicitly penalize poorly recovered detection efficiencies and guide the network towards accurate recovery of $\xi(\Lambda)$.

Prior to training, we randomly draw sets of compact binary parameters, $\{\theta_{I,j}\}$, from several different population models $p(\theta|\Lambda_I)$. Here, we use I to index population models and j to index the individual draws from a given population. Within each training step, we compute the detection efficiencies for these reference populations as predicted by the emulator in its current state, each of which can be approximated via the average

$$\hat{\xi}_I \approx \frac{1}{N_I}\sum_j^{N_I} \hat{P}(\det|\theta_{I,j}), \tag{5}$$

where N_I is the number of draws from population I. For each population, the detection efficiency can also be estimated via reweighting of the pipeline training injections. If $p(\theta|\Lambda_{\rm inj})$ is the distribution from which these injections were drawn and $N_{\rm total}$ is the total number of injections performed, then

$$\xi_I \approx \frac{1}{N_{\text{total}}} \sum_{\text{Found } i} \frac{p(\theta_i|\Lambda_I)}{p(\theta_i|\Lambda_{\text{inj}})}. \tag{6}$$

If we take ξ_I estimated in this manner as our target, then we expect the product $N_I \xi_I$ to be binomial distributed with mean $N_I \xi_I$ and standard deviation $\sqrt{N_I \xi_I (1-\xi_I)} \approx \sqrt{N_I \xi_I}$. In the limit of large N_I, the distribution can be well-approximated by a Gaussian, and the likelihood of $\hat{\xi}_I$ itself is

$$p(\hat{\xi}_I|\xi_I) \propto \exp\left[-\frac{(\hat{\xi}_I - \xi_I)^2}{2\xi_I/N_I}\right]. \tag{7}$$

Strictly, $\hat{\xi}_I$ and ξ_I are each noisy estimators of some unknown, underlying detection efficiency, over which we could marginalize. We find via direct evaluation that this procedure yields only small corrections to Eq. (7).

We use Eq. (7) to define an additional loss term

$$\begin{aligned} \mathcal{L}_\xi &= \sum_I - \ln p(\hat{\xi}_I|\xi_I, N_I) \\ &= \sum_I \frac{(\xi_I - \hat{\xi}_I)^2}{2\xi_I/N_I}. \end{aligned} \tag{8}$$

We employ four reference populations in this manner: the three distributions traced by the binary black hole, neutron star-black hole, and binary neutron star pipeline injections [33], and a fourth population designed to approximate the observed astrophysical population of binary black hole mergers. Further details are provided in Appendix B 2.

The total training loss is given by the sum of Eqs. (3) and (8).

C. Parametrization of compact binary mergers

We take as input a 13-dimensional description of a compact binary: component masses, component spins, distance, sky position, binary inclination, and polarization angle.[2] Exactly *how* these parameters are presented to the neural network, however, strongly affects network performance. We find that the best performance is achieved when providing the squared amplitudes

$$\begin{aligned} \mathcal{A}_+^2 &= \left(\frac{(\mathcal{M}_c^{\text{det}}/M_\odot)^{5/6}}{D_L/\text{Gpc}} \frac{1+\cos^2\iota}{2}\right)^2, \\ \mathcal{A}_\times^2 &= \left(\frac{(\mathcal{M}_c^{\text{det}}/M_\odot)^{5/6}}{D_L/\text{Gpc}} \cos\iota\right)^2 \end{aligned} \tag{9}$$

as direct inputs to the neural network; these determine, at leading post-Newtonian order, the expected signal-to-noise ratios in "plus" and "cross" polarizations for inspiral-dominated signals [e.g., [4,43]]. Here, $\mathcal{M}_c^{\text{det}}$ is the detector-frame binary chirp mass [related to the source-frame chirp mass by $\mathcal{M}_c^{\text{det}} = \mathcal{M}_c(1+z)$, where z is the source's redshift], D_L is the luminosity distance, and ι is the inclination angle between a binary's orbital angular momentum and our line of sight. We also find it advantageous to "overspecify" binary masses, providing the network with both detector-frame total masses and chirp masses, as well as both the standard and symmetric mass ratios (also known as "data augmentation" in machine learning settings). Although only two of these four parameters are needed to fully specify the component masses of a binary, different inspiral stages and waveform effects are more strongly governed by different combinations of these parameters. We expect that a neural network would therefore need to "learn" each of these parameters anyway, a step that is saved if we instead provide them directly.

In contrast, we do not provide the network with all six spin degrees of freedom, but instead compress spin information into several "effective" parameters that are expected to capture most inspiral and precessional dynamics. We include the standard effective inspiral spin [44–46],

$$\chi_{\text{eff}} = \frac{\chi_1 \cos\theta_1 + q\chi_2 \cos\theta_2}{1+q}, \tag{10}$$

as well as an analogous "asymmetric" spin parameter,

$$\chi_{\text{diff}} = \frac{\chi_1 \cos\theta_1 - \chi_2 \cos\theta_2}{2}, \tag{11}$$

where χ_i and θ_i are the dimensionless component spin magnitudes and polar tilt angles. Both χ_{eff} and χ_{diff} appear in the leading-order spin corrections to the gravitational-

TABLE I. Compact binary parameters provided to the neural network $P(\det|\theta)$ emulator.

Parameter	Definition
$\ln \mathcal{A}_+^2$	Log squared amp. of "+" polarization; see Eq. (9)
$\ln \mathcal{A}_\times^2$	Log squared amp. of "×" polarization; see Eq. (9)
$\mathcal{M}_c^{\text{det}}/M_\odot$	Detector-frame chirp mass
$M_{\text{tot}}^{\text{det}}/M_\odot$	Detector-frame total mass
D_L/Gpc	Luminosity distance
η	Reduced mass ratio: $m_1 m_2/M_{\text{tot}}^2$
q	Mass ratio: m_2/m_1 (where $m_2 \leq m_1$)
α	Right ascension
$\sin\delta$	Sine declination
$\lvert\cos\iota\rvert$	Absolute value of cosine inclination
$\sin\psi$	Sine of polarization angle
$\cos\psi$	Cosine of polarization angle
χ_{eff}	Effective inspiral spin
χ_{diff}	"Antisymmetric" inspiral spin; see Eq. (11)
$\chi_{\text{p}}^{\text{gen}}$	Generalized precessing spin; see Eq. (12)

[2]We neglect the time and phase of binary coalescence. The resulting $P(\det|\theta)$ computed by our emulator should therefore be regarded as a time- and phase-averaged detection probability, assuming uniform distributions for each quantity.

wave inspiral phase entering at 1.5PN order [47]. We also include the generalized precessing spin parameter [48],

$$\chi_{\mathrm{p}}^{\mathrm{gen}} = [(\chi_1 \sin\theta_1)^2 + (\tilde{\Omega}\chi_2 \sin\theta_2)^2 + 2\tilde{\Omega}\chi_1\chi_2 \sin\theta_1 \sin\theta_2 \cos\Delta\phi]^{1/2}, \tag{12}$$

an extension of the precessing spin parameter first introduced in Ref. [49]. Here, $\Delta\phi$ is the angle subtended by the two component spins, once projected onto the plane perpendicular to the orbital angular momentum, and $\tilde{\Omega} = q(3+4q)/(4+3q)$.

The complete set of parameters passed to the neural network is listed in Table I.

III. PERFORMANCE OF THE TRAINED EMULATOR

Figure 3 demonstrates the performance of the trained network using binary black hole signals. The blue histograms show the distribution of found events among the binary black hole pipeline injections (see the left-hand panel of Fig. 2) [33]. The underlying population from which these injections were drawn is indicated via the dotted histograms; this distribution is described further in Appendix B 1. To test the accuracy with which our neural network emulates $P(\det|\theta)$, we draw a new set of proposed binary black holes from this same underlying distribution. We use our trained network to assign detection probabilities $\hat{P}(\det|\theta)$ to each of these systems; these are then rejection sampled in accordance with the predicted probabilities to randomly identify a subset as successfully detected. This process is repeated until we gather 10^4 new detections. The resulting distributions of found events are shown via the solid black histograms. Figures 4 and 5 analogously show the reconstructions of detected binary neutron star and neutron star-black hole binary populations.

The detected distributions of each class of compact binary, as predicted by the neural network emulator, are good visual matches to the distributions of compact binaries actually detected by search pipelines. As a more quantitative metric, we compute Kolmogorov-Smirnov test statistics between actual (blue) and emulated (solid black) distributions of found injections; the p-values of these test statistics are shown in the upper-right corner of each

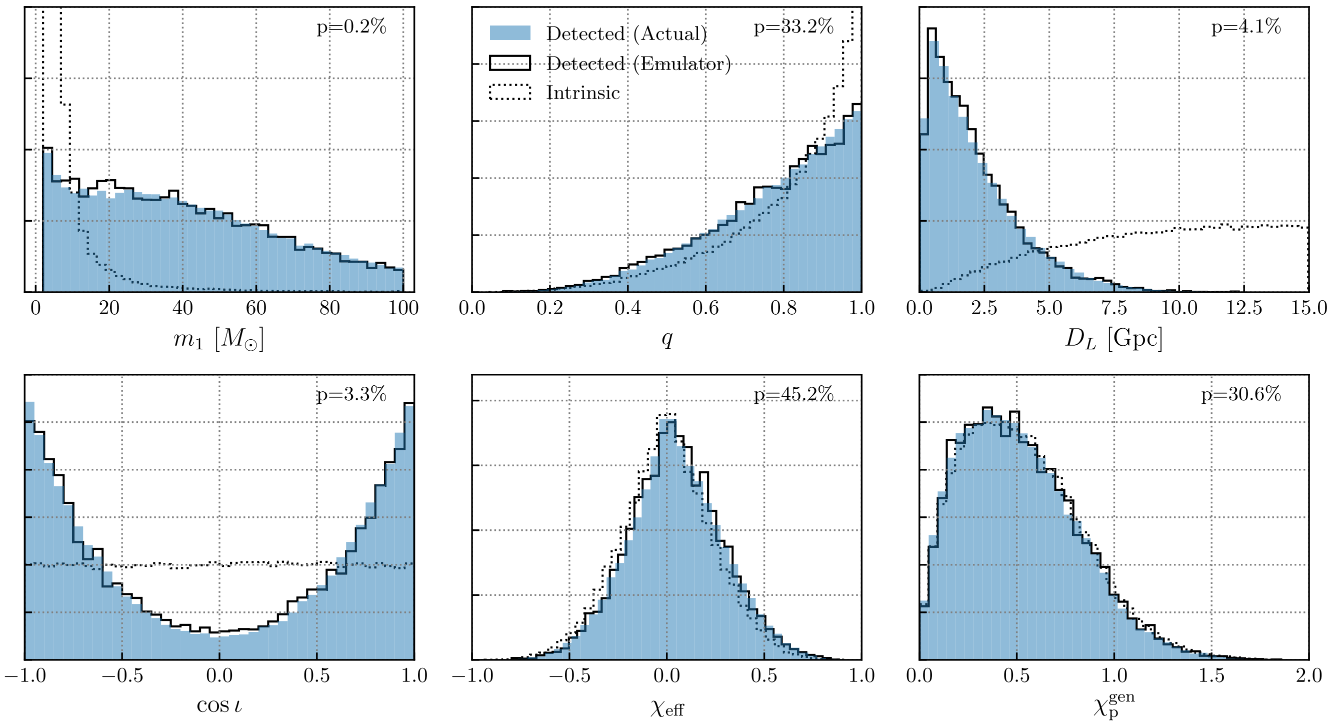

FIG. 3. Distribution of detected binary black hole mergers as predicted by our trained neural network $P(\det|\theta)$ emulator (solid black distributions), compared to the distributions of found pipeline injections (found pipeline injections). Both populations are drawn from identical intrinsic distributions (dotted black). The trained neural network emulator produces distributions of found binary black holes that are near matches to the actual distribution of found events from compact binary search pipelines. The numbers inset in the upper-right corner of each plot show p-values of Kolmogorov-Smirnov test statistics between the emulated and actual distributions of found events. These p-values indicate good statistical agreement between most pairs of distributions, but also that some pairs are not formally indistinguishable. The emulated and actual distributions of m_1 values, for instance, have a $p = 2\times10^{-3}$ probability of being drawn from the same parent distribution.

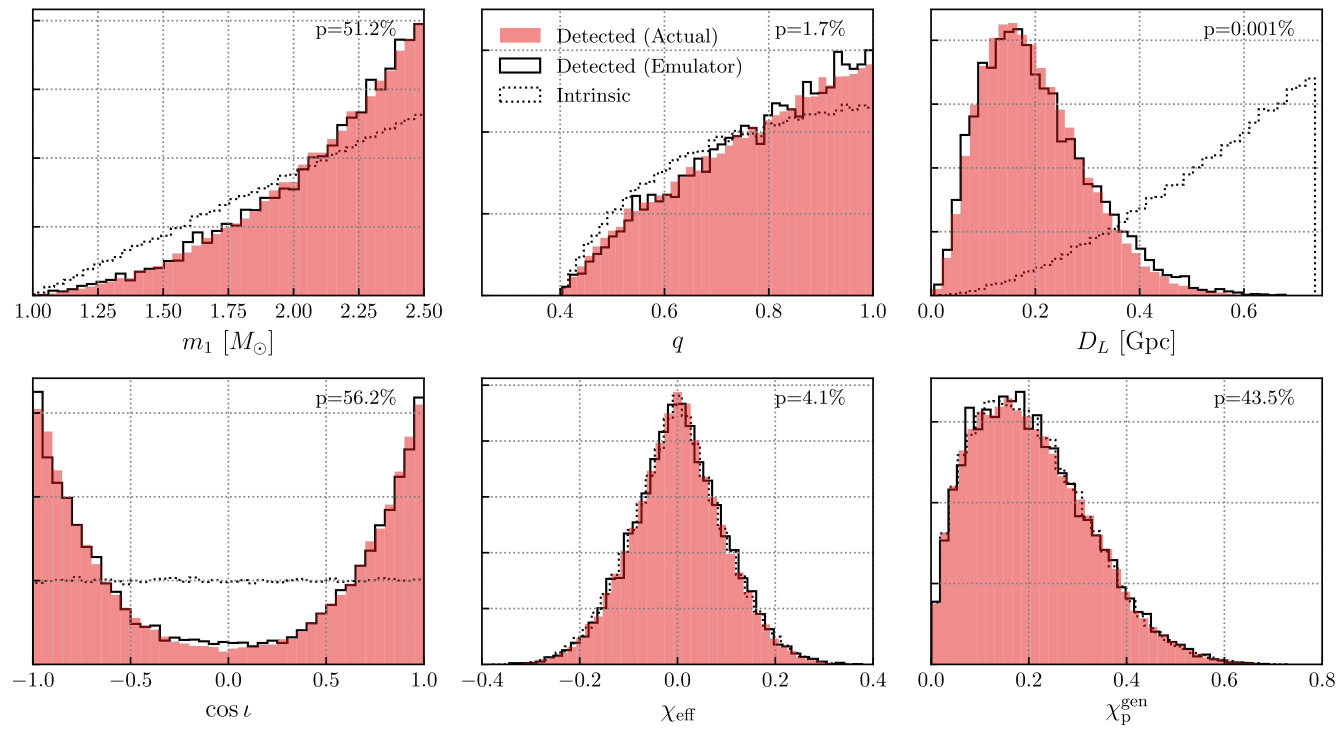

FIG. 4. As in Fig. 3 but for the population of detectable binary neutron stars.

subplot. Most p-values lie above 10^{-2}, indicating good statistical agreement. Some p-values are lower, though. The actual and emulated distributions of binary neutron star distances, for example, have only a 10^{-5} chance of being drawn from the same parent distribution.

A well-working $P(\det|\theta)$ emulator not only should produce the correct distributions of detected compact binary parameters, but also must reproduce the correct *absolute fraction* of events that are successfully detected (the former does not necessarily imply the latter).

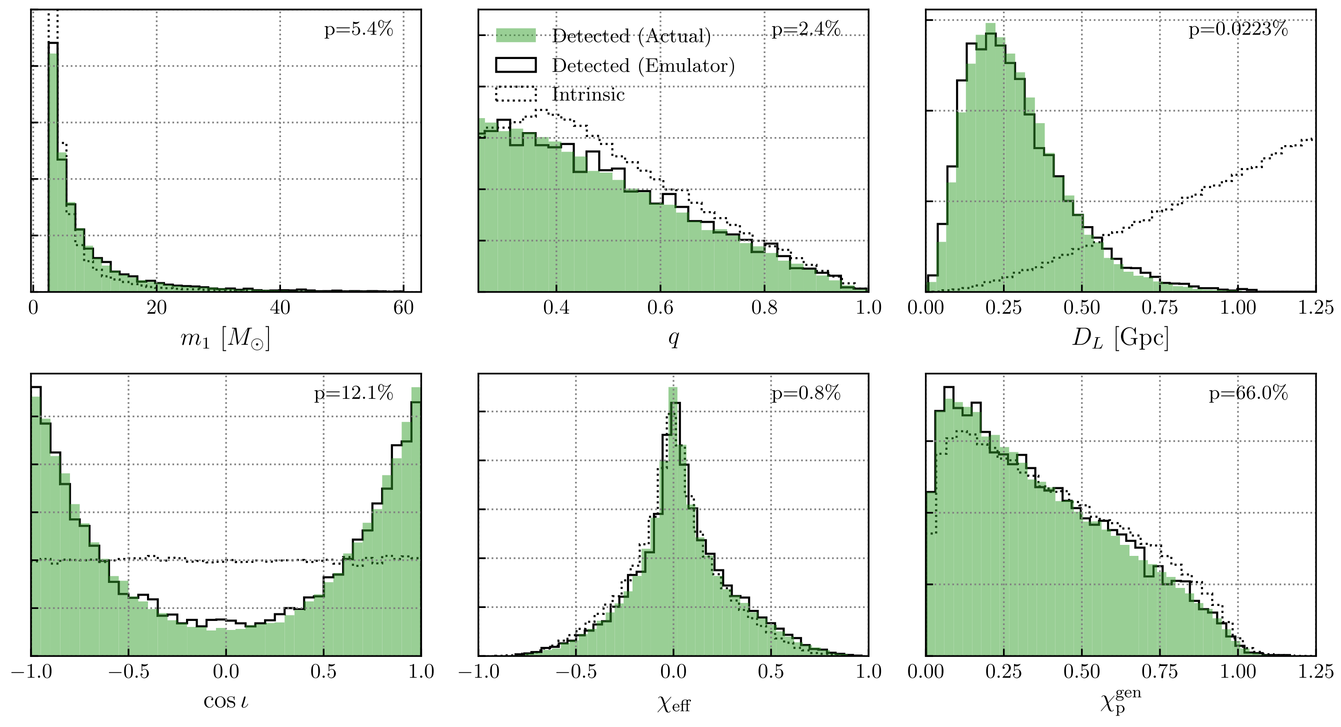

FIG. 5. As in Fig. 3 but for the population of detectable neutron star-black hole binaries.

The absolute detection efficiency ξ, defined above in Eq. (4), is necessary to successfully predict gravitational-wave detection rates and is a critical ingredient in the statistical inference of astrophysical compact binary populations. To test the ability of the trained emulator to produce accurate detection efficiencies, we repeatedly and randomly draw from a large space of possible binary black hole populations. Primary masses are assumed to follow a superposition between a power law and a Gaussian peak, secondary masses are power-law distributed, spin magnitudes and spin-orbit misalignment angles follow truncated Gaussians, and the merger rate is assumed to grow as a power law in $1+z$ (see Appendix C for these exact distributions and the range of hyperparameters chosen). For each proposed population, we then compute the integrated detection efficiency $\xi(\Lambda)$ in two ways. First, we estimate $\xi(\Lambda)$ via standard reweighting of the found binary black hole pipeline injections, as in Eq. (6) above. Second, we instead compute the detection efficiency using our trained neural network, drawing an ensemble of binary parameters $\{\theta\} \sim p(\theta|\Lambda)$ from the proposed population and then directly evaluating the detection efficiency as in Eq. (5).

Figure 6 shows the resulting detection efficiencies computed in both manners. Each point corresponds to a randomly chosen population, and error bars correspond to expected Poisson uncertainties given the finite number of pipeline injections/draws used for each efficiency calculation. In general, we see good agreement between $\xi(\Lambda)$ values obtained through traditional injection reweighting and values computed with our $P(\det|\theta)$ emulator across efficiencies spanning many orders of magnitude. We do note that there are several points for which the two methods disagree, with the neural network predicting noticeably higher detection efficiencies than the reweighted pipeline injections. Each of these points, though, has significant uncertainty in the reweighted pipeline injections' efficiency calculation, corresponding to populations that are very different from the reference distribution by which pipeline injections were drawn (specifically, these populations strongly favor unequal-mass binaries).

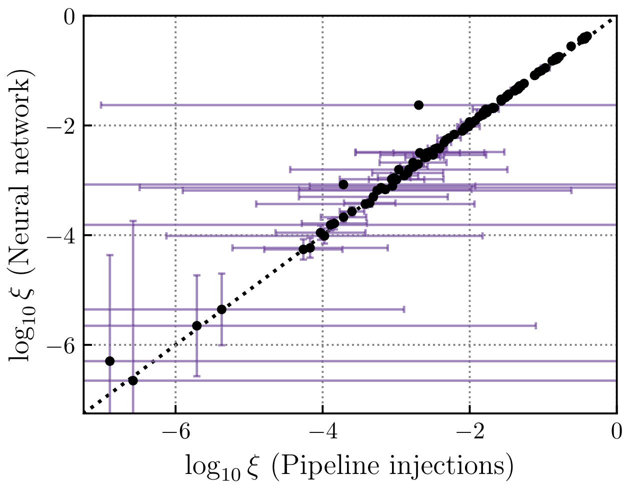

FIG. 6. Comparison of integrated binary black hole detection efficiencies as computed by traditional reweighting of pipeline injections (x-axis) vs our trained neural network emulator (y-axis). Each point represents a different possible binary black hole population, with randomly chosen mass, mass ratio, spin, and redshift distributions. The error bars show expected uncertainties, given the finite number of pipeline injections/population draws informing each estimate. The large majority of proposed populations yield efficiency estimates that are consistent between the two methods. There exist a small number of outlier points, for which the methods do not agree; these correspond to populations for which the pipeline injection-based efficiency estimates are highly uncertain, due to poor reweighting efficiencies from the injections' parent distribution.

IV. STABILIZING HIERARCHICAL INFERENCE

Hierarchical inference of the compact binary population can be limited by the accuracy with which the detection efficiency $\xi(\Lambda)$ can be estimated. The detection efficiency is most commonly computed by reweighting a fixed set of pipeline injections, as in Eq. (6). Accurate estimation of $\xi(\Lambda)$ in this manner requires that (i) the injections were drawn from a parent distribution, $p(\theta|\Lambda_{\rm inj})$, that is "close to" the target population $p(\theta|\Lambda)$, and/or (ii) that a very large number of found injections be available, which in turn requires a very large number of total trials $N_{\rm total}$.

It is not clear what, formally, is meant by "close to" in the previous sentence. What is clear, though, is that one or both of the above conditions can readily fail in practice, producing imprecise estimates of $\xi(\Lambda)$ and hampering inference of the compact binary population. A common diagnostic is the "effective number" of injections informing an estimate of $\xi(\Lambda)$. If we define $w_i = p(\theta_i|\Lambda)/p(\theta_i|\Lambda_{\rm inj})$ as a short-hand for the ratio appearing in Eq. (6), then the number of effective injections is

$$N_{\rm eff} = \frac{(\sum_i w_i)^2}{\sum_j w_j^2}, \tag{13}$$

where both sums are again taken over found injections. In order for systematic uncertainty in $\xi(\Lambda)$ due to a finite number of injections to remain a subdominant effect, it has been argued that one requires $N_{\rm eff} \gg c N_{\rm events}$, where $N_{\rm events}$ is the number of observed compact binaries and c is some constant that is (hopefully) of order unity [30,32].[3] This implies that the total number of pipeline injections must scale linearly with catalog sizes, such that $N_{\rm total} \propto N_{\rm events}$. However, other authors have argued that the number of injections must scale more steeply with

[3]In practice, a common choice is to demand that $N_{\rm eff} \geq 4N_{\rm events}$, following Ref. [30].

catalog size, such that $N_{\mathrm{total}} \propto N_{\mathrm{events}}^{\alpha}$ with $1.5 \lesssim \alpha \lesssim 2$ [35]. Poorly converged $\xi(\Lambda)$ estimates due to insufficient injections already and not infrequently limit our ability to explore the compact binary population. The required increase of pipeline injections with catalog size (regardless of the precise scaling) implies that this issue will persist or be further exacerbated in the future.

A. Dynamically drawing injections

A trained $P(\det|\theta)$ offers one avenue to mitigating the problem of poor $\xi(\Lambda)$ estimation. The central problem, and the reason pipeline injections must grow in number with catalog size, is the fact that we typically must make do with a fixed injection set. With a trained emulator, we can abandon this constraint and instead *dynamically draw new injections* from each new population of interest. A particularly efficient algorithm for doing this in the context of hierarchical inference is the following [50]:

(1) For each compact binary parameter, draw a large number of random values on the interval [0, 1]:

$$\begin{aligned} \boldsymbol{c}_{m_1} &\sim U(0,1) \\ \boldsymbol{c}_{m_2} &\sim U(0,1) \\ \boldsymbol{c}_{z} &\sim U(0,1), \\ &\text{etc.}, \end{aligned} \tag{14}$$

where, e.g., $\boldsymbol{c}_{m_1} \equiv \{c_{m_1}\}$ indicates a vector of individual draws. This is done once, prior to beginning inference. To improve convergence, these random values can be sampled jointly via low-discrepancy sequences, such as the Sobol sequence [51].

(2) Proceed with inference. Within the first likelihood evaluation with some proposed population Λ, compute, analytically or numerically, the inverse cumulative distribution function $F_{\Lambda}^{-1}(\cdot)$ associated with each compact binary parameter.

(3) Apply these inverse distributions to our draws from the unit interval to yield sets of physical parameter values:

$$\begin{aligned} \boldsymbol{m}_1 &= F_{\Lambda,m_1}^{-1}(\boldsymbol{c}_{m_1}) \\ \boldsymbol{m}_2 &= F_{\Lambda,m_2}^{-1}(\boldsymbol{c}_{m_2}) \\ \boldsymbol{z} &= F_{\Lambda,z}^{-1}(\boldsymbol{c}_{z}). \end{aligned} \tag{15}$$

The resulting values will be distributed according to the proposed population density, $p(\theta|\Lambda)$.

(4) Assemble these into a matrix $\boldsymbol{\Theta} = (\boldsymbol{m}_1 \boldsymbol{m}_2 \boldsymbol{z} \ldots)$ of compact binary parameters with shape $(N_{\mathrm{samp}}, N_{\mathrm{dim}})$, and evaluate their detection probabilities with the trained neural network, yielding a vector of detection probabilities $\boldsymbol{P} = \hat{P}(\det|\boldsymbol{\Theta})$ with length N_{samp}.

(5) Take the mean of the N_{samp} samples in $\boldsymbol{P}$ to obtain the detection efficiency: $\xi(\Lambda) = \langle \boldsymbol{P} \rangle$.

(6) Repeat Steps 2–5 for each subsequent likelihood evaluation.

The scheme outlined above assumes a factorizable population model, such that the joint distribution $p(m_1, m_2, z, \ldots|\Lambda)$ can be written as the product $p(m_1|\Lambda)p(m_2|\Lambda)p(z|\Lambda)\ldots$. It can, however, be straightforwardly extended to nonfactorizable populations with intrinsic correlations between parameters. In this case, one iteratively performs inverse transform sampling using conditional probability distributions. For example:

(1) Compute the cumulative probability distribution $F_{\Lambda,z}(z)$ of source redshifts, and inverse transform sample to obtain a redshift z drawn from $p(z|\Lambda)$.

(2) Given this redshift sample, define the conditional primary mass distribution $p(m_1|z,\Lambda)$. Compute the cumulative distribution of this conditional distribution, and inverse transform sample to obtain a primary mass drawn from $p(m_1|z,\Lambda)$.

(3) etc.

The result will be a tuple $\{z, m_1, m_2, \ldots\}$ drawn from the joint distribution $p(m_1, m_2, z, \ldots|\Lambda)$.

Because the detection efficiency is evaluated using draws directly from the population Λ of interest, the above algorithm can enable more accurate estimation of $\xi(\Lambda)$ than can be obtained through reweighting of fixed injections. This is particularly true when attempting to investigate *narrow* population features. Narrow or abrupt features in the compact binary population are often of great astrophysical interest but are notoriously difficult to study, computationally speaking [e.g. [52]]. This is due, in part, to the fact that an estimate of $\xi(\Lambda)$ via reweighting of fixed injections will be necessarily be dominated by the small number of injections that happen to lie in the immediate vicinity of the feature of interest. The resulting $\xi(\Lambda)$ will be subject to a small effective sample count and/or yield large log-likelihood variance.

The above algorithm, enabled by our $P(\det|\theta)$ emulator, avoids the poor convergence of $\xi(\Lambda)$ in the presence of narrow population features. As a demonstration, the upper panel of Fig. 7 shows estimates of the detection efficiency for an observationally plausible binary black hole population (see Appendix C for details) as we vary the assumed width σ_χ of the component spin magnitude distribution. The purple curve shows the detection efficiency as calculated using our neural network emulator, following the algorithm above, while the green curve shows values obtained through reweighting of fixed pipeline injections. The lower panel, meanwhile, shows the effective number of injections [Eq. (13)] informing these estimates.

When σ_χ is large and the component spin distribution is broad, all is well: both methods yield nearly equal $\xi(\Lambda)$ values, and each is informed by a large number of effective samples, signifying that these values are robust. As σ_χ is

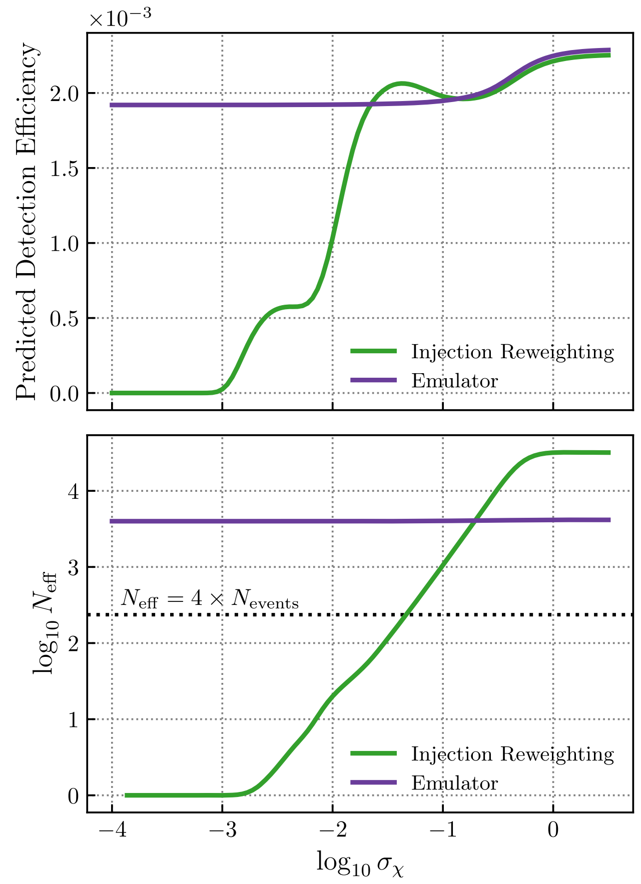

FIG. 7. Illustration of convergence issues mitigated by use of a $P(\det|\theta)$ emulator in hierarchical inference. Top: predicted binary black hole detection efficiency ξ as a function of the presumed width of the astrophysical spin magnitude distribution. Detection efficiencies are calculated via the reweighting of fixed pipeline injections (green) and by using the trained $P(\det|\theta)$ emulator to draw new injections at each value of σ_χ following the algorithm in Sec. IV A (purple). At large σ_χ, both methods predict comparable detection efficiencies. As σ_χ decreases, however, the efficiencies predicted by injection reweighting drop unphysically to zero, due to the lack of injections falling inside the narrow range supported by the population model. Bottom: the effective number of injections [see Eq. (13)] informing each estimate. As σ_χ becomes small, the number of informative injections approaches zero, driving the unphysical behavior in the top panel. In particular, when $\sigma_\chi \lesssim 0.03$ we fail the convergence criteria $N_{\rm eff} \geq 4 \times N_{\rm events}$ commonly adopted in the literature. When instead using the $P(\det|\theta)$ to draw new injections from each new distribution of interest, the number of effective injections remains approximately constant and the estimate of ξ well-converged.

lowered and the spin distribution narrows, however, the injection reweighting begins to exhibit problems. Within the lower panel, we see that the effective number of injections drops quickly; by the time we reach $\log_{10}\sigma \approx -1.5$, we are already falling below the $N_{\rm eff} \geq N_{\rm events}$ threshold often adopted for reliable inference. For even smaller σ_χ we see the estimation of $\xi(\Lambda)$ break down. The inferred detection efficiency rises briefly before plummeting unphysically to zero. In contrast, direct evaluation with the neural network emulator remains well behaved, even as the component spin magnitude distribution approaches a delta function.

In principle, this approach is possible for semianalytic sensitivity estimates as well. For example, Essick [34] provides a closed-form estimate for $P(\det|\rho_{\rm opt})$ that accounts for the probability of different noise realizations. However, in practice, this approach would require many waveform calls within each likelihood evaluation, which would be very costly. The neural emulator avoids this by directly learning $P(\det|\theta)$ instead of $P(\det|\rho_{\rm opt})$.

Another advantage of the algorithm above is that it is *differentiable*. By drawing a fixed set of random values from the unit intervals (Step 1) and later inverse transforming sampling to obtain physical parameter values, we ensure that the likelihood is a deterministic function of Λ and hence amenable to algorithms like Hamiltonian Monte Carlo.

We caution that the use of a detection probability emulator as described in this section is still subject to Monte Carlo variance. Different realizations of random values in Eq. (14) will, in turn, yield slightly different estimated detection efficiencies. This variance will decrease as one increases the number of Monte Carlo samples, but in some cases the required number of samples may be large, particularly if binary detections come primarily from a very small (and hence improbable) portion of parameter space. For example, the integrated detection efficiency is usually dominated by the small fraction of events at low redshift, whereas the vast majority of events under reasonable population models occur at high redshifts; a very large number of samples will therefore be needed to obtain a reasonable number of nearby events. In such cases, one can instead adopt a variant of the algorithm described above, in which some parameters are inverse-transform sampled directly from the proposed population $p(\theta|\Lambda)$ while others (like redshift) are reweighted from a fixed reference distribution. Such a hybrid strategy can improve convergence and decrease the overall number of samples required to estimate $\xi(\Lambda)$. More details are provided in Appendix D.

B. Full hierarchical inference: A demonstration

As a further demonstration of this approach, as well as a test of our $P(\det|\theta)$ emulator, we perform complete hierarchical inference of the binary black hole population using the algorithm described above to compute $\xi(\Lambda)$. We use the 59 binary black holes observed during the LIGO-Virgo O3 observing run with false-alarm rates below 1 yr^{-1} [41,42],[4] and adopt population models comparable to those used in recent LIGO-Virgo-KAGRA Collaboration

[4]The events GW190814 [53] and GW190917 [54], with their very uneven mass ratios and low secondary masses, are excluded as outliers relative to the bulk binary black hole population [19].

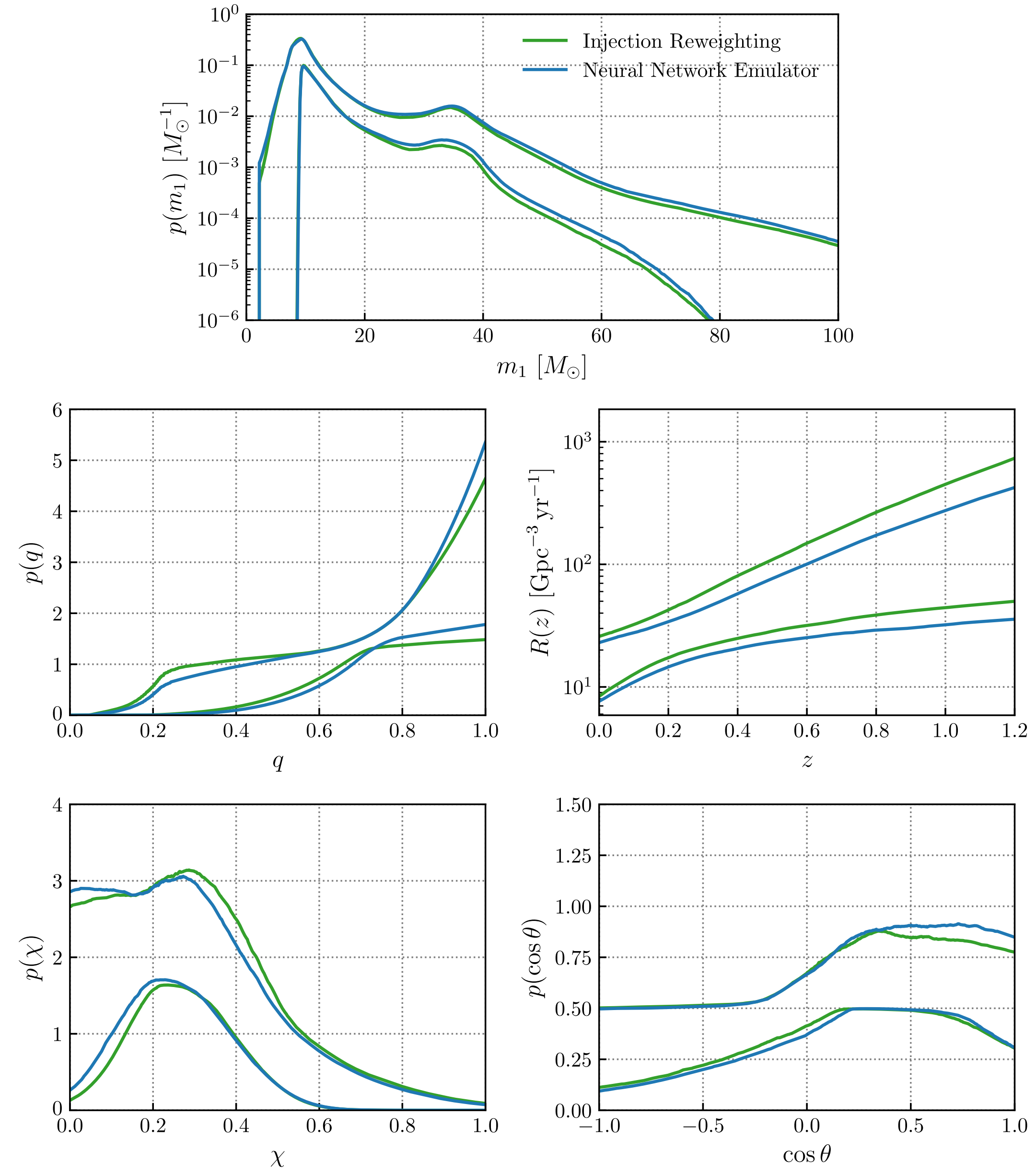

FIG. 8. Measurements of the binary black hole population, using the 59 binary black holes detected in the LIGO-Virgo-KAGRA O3 observing run with false-alarm rates below 1 yr^{-1}. Green curves show the central 95% credible regions when correcting for selection effects via reweighting the fixed set of pipeline injections spanning the O3 observing run. Blue curves, meanwhile, show results obtained when instead using our trained $P(\det|\theta)$ emulator, along with the algorithm described in Sec. IV A, to dynamically draw *new* found injections directly from the proposed population within each likelihood evaluation. The underlying posteriors on the hyperparameters governing the binary black hole mass, spin, and redshift distributions are shown in Appendix E; see Figs. 14–16.

analyses [19]. Specifically, we assume that source-frame primary masses follow a mixture between a power-law continuum and a Gaussian peak [55] and that secondary masses are power-law distributed [56]. Component spin magnitudes follow a truncated Gaussian distribution, while cosine spin-orbit tilts are described as a mixture between an isotropic component and a Gaussian excess [17]. The source-frame binary black hole volumetric merger rate is assumed to follow a power law in $(1+z)$ [36]. The exact population models used and the priors on their parameters are presented in Appendix C.

Results are shown in Fig. 8. The pair of green curves shows the 95% credible constraints on the probability distribution/merger rates of binary black holes using standard injection reweighting. Blue curves show constraints instead obtained using the trained $P(\det|\theta)$ emulator. Both sets of results are near matches, with the emulator yielding accurate reconstruction of the binary black hole primary mass, mass ratio, spin magnitude, and spin tilt distributions, as well as accurate reconstruction of the redshift-dependent merger rate. Posteriors on the underlying hyperparameters under both approaches can be seen in Appendix D.

While the results in Fig. 8 are consistent with one another, they are not exact matches. In particular, the neural-network-based selection effects yield a slightly stronger preference for equal mass ratios and slightly less evolution of the merger rate with redshift, relative to selection effects estimated via injection reweighting. It is not clear which approach is more accurate. On the one hand, the neural network emulator may be more reliably interpolating the underlying selection function, particularly in regions where injections are sparse (such as low mass ratios). On the other hand, because the neural network was trained on the same injections informing the results in green, we might expect a perfectly performing emulator to yield identical results (up to variance associated with Monte Carlo averaging). The slight differences in Fig. 8 may therefore indicate further room for improvement.

V. COMPARISON TO SEMIANALYTIC SELECTION EFFECTS

As noted in Sec. I, it is common for gravitational-wave selection effects to be semianalytically approximated via a threshold on a source's matched filter SNR. This threshold is often placed on a source's optimal SNR or on a simulated realization of an "observed" SNR that mimics random fluctuations due to noise. Although SNR thresholds can approximate search selection effects, they are known to neglect higher order aspects of gravitational-wave detection, including signal consistency checks and nonstationary noise. In this section, we compare our trained $P(\det|\theta)$ emulator to traditional semianalytic SNR thresholds, exploring the degree to which the neural network learns additional, higher order information not contained in SNRs alone.

In Figure 9, we again show the distribution of successfully detected binary black hole injections (blue) together with predictions from our trained neural network emulator (solid black); see Fig. 3. Each panel of this figure contains two additional curves. The dotted histograms show properties of found events as predicted by a semianalytic threshold on a source's optimal SNRs. Specifically, an ensemble of simulated events is drawn from the same parent distribution as the real pipeline injections. Each simulated event is placed at a random time during the O3 observing run, and its optimal matched filter SNR is computed, summing in quadrature over the three Advanced LIGO and Advanced Virgo detectors. Sources are labeled as "detected" if their optimal SNRs exceed $\rho_{\mathrm{opt}} \geq 10$, a threshold found to broadly approximate the 1 yr^{-1} false-alarm rate threshold adopted in this and many other works. The dashed histograms are analogous, but constructed by instead demanding that simulated "observed" network SNRs exceed $\hat{\rho} \geq 10$, following Ref. [34]. These observed SNRs are randomly drawn from the probability distribution of possible SNRs for each event, accounting for the effects of random noise fluctuations.[5]

Within Fig. 9, we see that, while our trained $P(\det|\theta)$ emulator yields the correct distribution of effective precessing spin parameters, the semianalytic SNR thresholds do not. This may reflect the fact that matched filtering template banks do not generally include effects of spin precession, and thus semianalytic calculations may systematically overestimate the SNRs of strongly-precessing binaries. Barring χ_{p}, however, it appears that the semianalytic thresholds on both optimal and observed SNRs broadly recover realistic distributions of detected binary black holes, performing comparably to our trained emulator.

This conclusion breaks down, however, if we more carefully consider predicted detections as a function of distance. Figure 10 shows, in blue, the cumulative probability distribution of found pipeline injections within three consecutive luminosity distance shells. Also shown are predictions from the neural network emulator (solid black), the semianalytic optimal SNR cut (dotted black), and the semianalytic observed SNR cut (dashed black). As we move to larger distances, we see that a semianalytic threshold on optimal SNRs predicts binary black hole detections systematically shifted towards larger masses. In other words, this strategy *systematically underestimates* the sensitivity of Advanced LIGO and Advanced Virgo detectors to low-mass binaries and *systematically overestimates* the sensitivity to high-mass binaries. This bias is lessened by instead thresholding on simulated observed SNRs, but it remains present. This behavior is consistent with Ref. [34], which found that a *mass-dependent* SNR threshold was needed to accurately predict the distance distribution of detected binary black holes, with higher-mass binaries requiring higher semianalytic thresholds (see their Fig. 8).

The trained neural network, in contrast, produces accurate cumulative distributions in all distance bins (the elevated variance in the 8–12 Gpc interval is due to a very small number of events in this range). We therefore conclude that the $P(\det|\theta)$ emulator is successfully learning higher-order information encoded in pipeline injections but not captured by a simple SNR threshold.

We show analogous results for binary neutron stars and neutron star-black hole binaries in Figs. 11 and 12, respectively. The trends identified above for binary black holes persist: semianalytic SNR thresholds tend to poorly predict effective precessing spin distributions and underpredict the distances to which low-mass systems are successfully detected, while the neural network emulator more accurately matches found pipeline injections. Within Fig. 12, we also see that semianalytic approximations overpredict the sensitivity of Advanced LIGO and

[5]The quantity $\hat{\rho}$ is referred to as $\rho_{\mathrm{net},\phi}$ in Ref. [34].

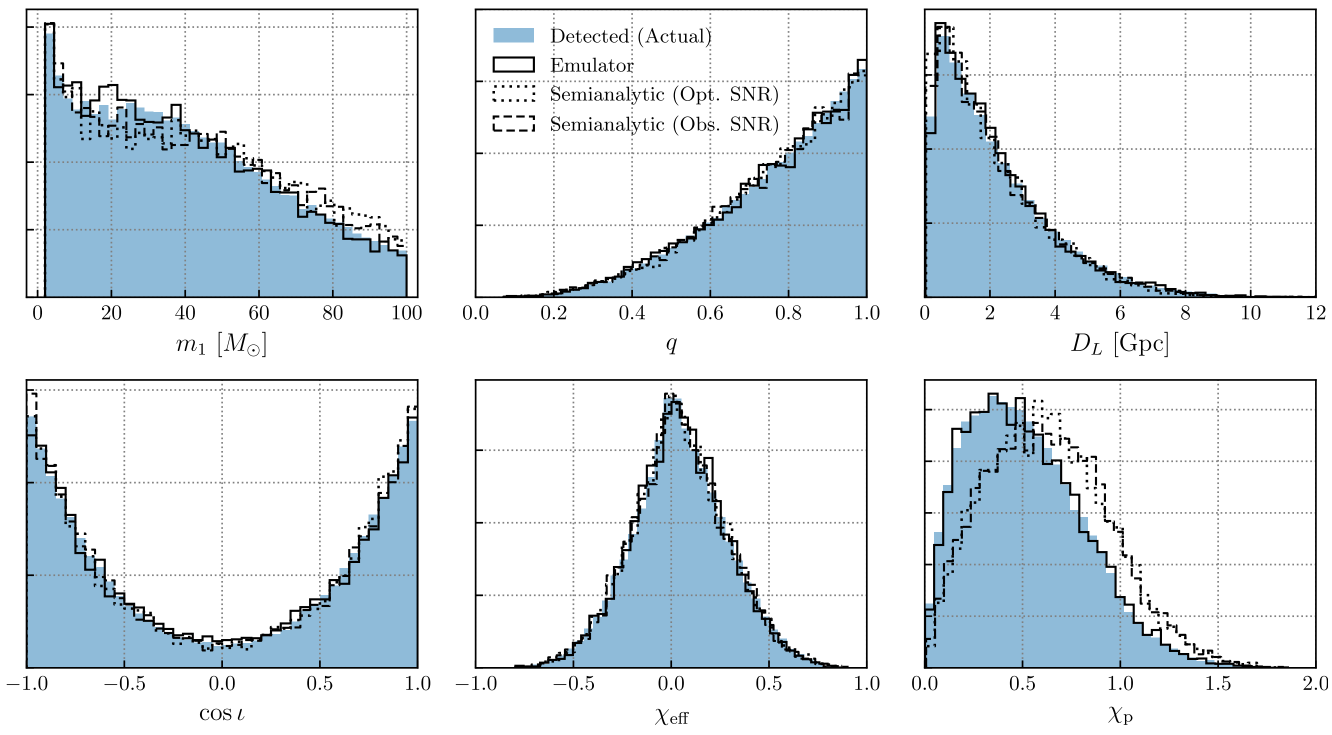

FIG. 9. As in Fig. 3, but additionally detected binary black hole properties predicted by approximating Advanced LIGO and Advanced Virgo selection effects via a threshold on matched filter signal-to-noise ratio. Dotted distributions show results when demanding an optimal SNR $\rho_{\rm opt} \geq 10$, while dashed distributions correspond to a threshold $\hat{\rho} \geq 10$ on randomly-perturbed "observed" SNRs that capture the effects of fluctuating noise [34]. With the exception of the effective precessing spin parameter, semianalytic SNR thresholds can broadly capture the distributions of detectable binary black holes, with visual matches comparable to our trained neural network emulator. However, we see that SNR thresholds systematically underestimate the sensitivity of the LIGO-Virgo network to low-mass binaries at large distances, whereas the neural network does not; see Fig. 10.

Advanced Virgo detectors to systems with very unequal masses, with sensitivity more accurately captured by the $P(\det|\theta)$ emulator.

We note that, although (possibly parameter-dependent) semianalytic SNR thresholds and our neural network emulator may successfully predict similar distributions of compact binary detections, SNR thresholds *cannot* compare with the neural network emulator in hierarchical inference. As described in Sec. IV, the utility of the neural network is its ability to estimate $P(\det|\theta)$ for a new ensemble of binaries in each new likelihood evaluation. This, in turn, requires a $P(\det|\theta)$ that is computationally

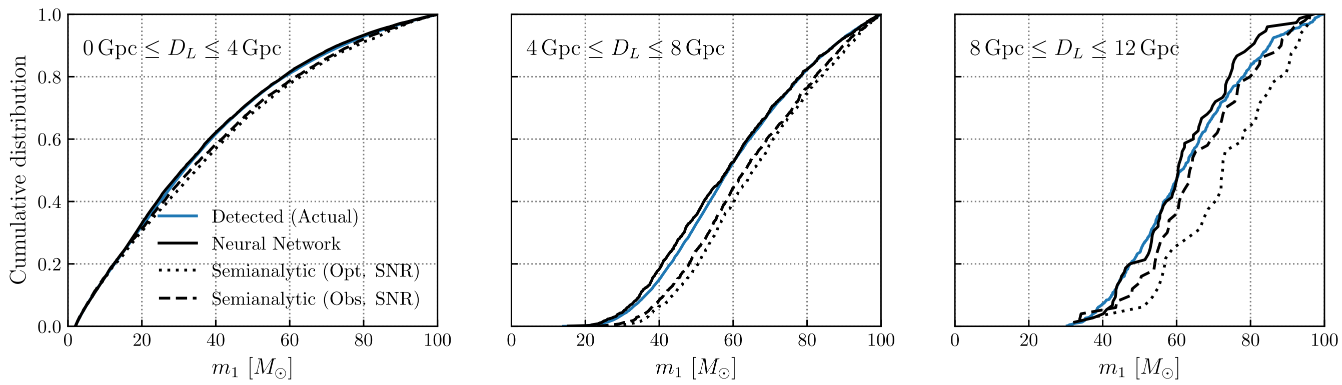

FIG. 10. Cumulative distributions of detected binary black hole primary masses in three different luminosity distance intervals, as predicted by real pipeline injections (blue), the trained $P(\det|\theta)$ emulator (solid black), and semianalytic thresholds on optimal and observed SNRs (dotted and dashed black, respectively). In all distance intervals, semianalytic sensitivity estimates underestimate the detectability of low-mass binaries, shifting predicted cumulative distributions to the right.

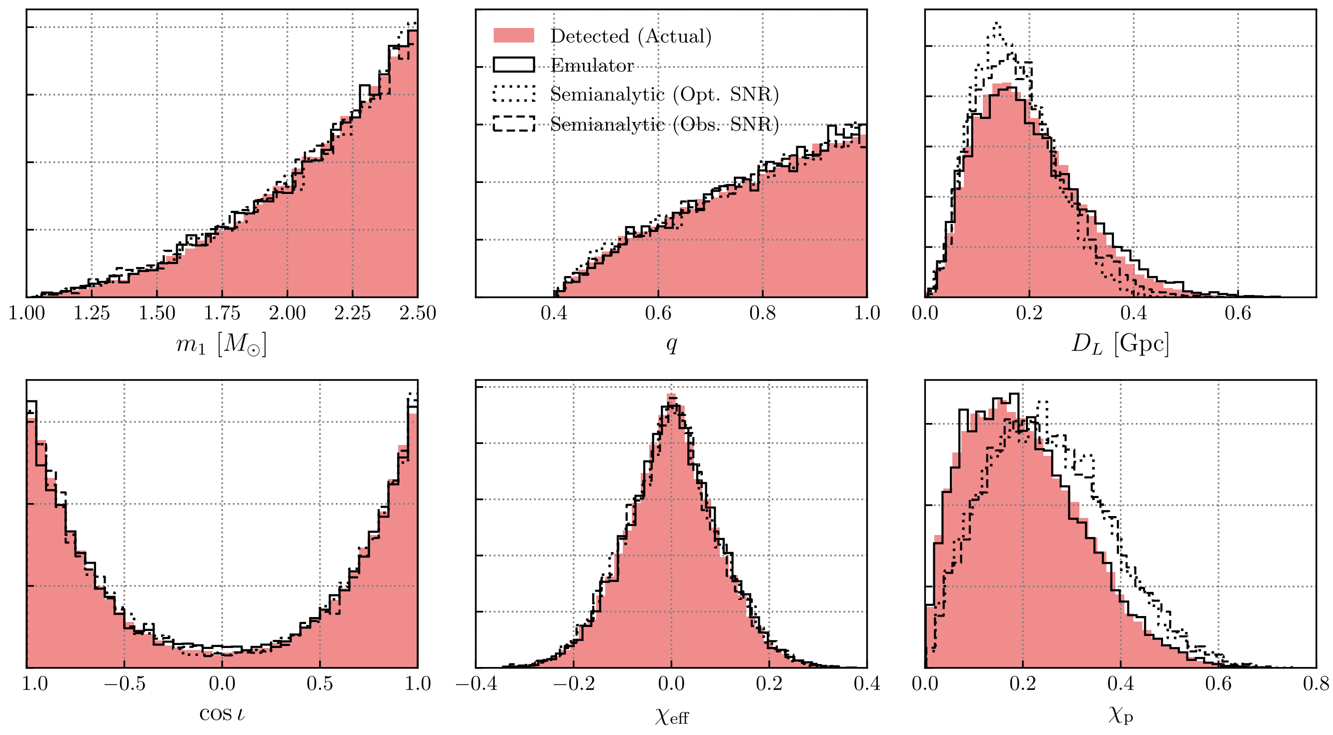

FIG. 11. As in Fig. 9, but for binary neutron stars.

efficient and, in modern computing environments, differentiable. Detection probability estimation via SNR thresholding satisfies neither of these requirements. In particular, the calculation of signal-to-noise ratios requires the evaluation of gravitational waveforms; waveform generation is usually slow and non-differentiable, and thus cannot be used in an algorithm like that described in Sec. IVA.

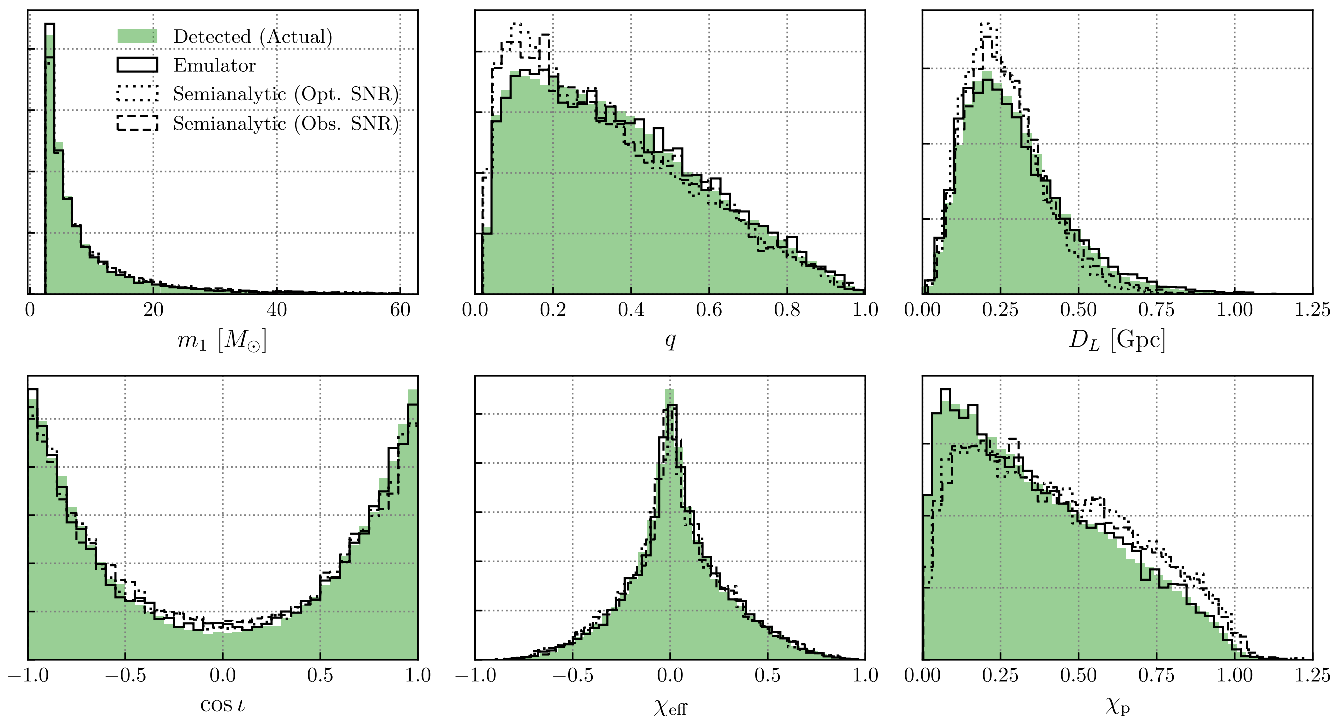

FIG. 12. As in Fig. 9, but for neutron star-black hole binaries.

VI. DISCUSSION

In this paper we have presented a neural network emulator with which to describe the detection probabilities of compact binary mergers in the LIGO-Virgo-KAGRA O3 observing run. We described the construction and training of the emulator, and validated its accuracy via complete hierarchical inference of the binary black hole population.

This is not the first such tool in the literature; a number of previous studies have pursued various machine learning strategies to model gravitational-wave selection effects [38,57–60]. Reference [38] trained a neural network to classify compact binaries as detectable or undetectable. Like our study, this classification was performed on a per event basis. One limitation of that work, however, was its reliance on an idealized detection process: Binaries were regarded as "detectable" if they exceeded an optimal matched filtering signal-to-noise ratio threshold, calculated using fixed instrumental noise power spectral densities. In reality, drifting instrumental sensitivities, terrestrial noise transients, and additional signal consistency requirements mean that compact binary detection is more complicated than a universal signal-to-noise ratio threshold. While the emulators presented in Ref. [38] are likely sufficient for qualitatively accurate forecasting of future detections, they likely do not capture these higher order effects required for precise inference of the binary population; for this, injection campaigns (like those comprising our training data) are needed.

A complementary strategy was undertaken in Ref. [57], in which $P(\det|\theta)$ emulation is framed as a problem of *density estimation*. In this approach, a Gaussian mixture model was fit to the distribution of found injections, yielding a function proportional to the product $P(\det|\theta)p(\theta|\Lambda_{\mathrm{inj}})$, where $p(\theta|\Lambda_{\mathrm{inj}})$ is the distribution from which the injections were drawn. Relative detection probabilities of *new* injections θ' could then be calculated by evaluating the mixture model and dividing by $p(\theta'|\Lambda_{\mathrm{inj}})$, leaving values proportional to $P(\det|\theta')$. This density estimation approach has the advantage of being computationally efficient and inexpensive to train. One disadvantage is the requirement that users must track and evaluate the draw probabilities $p(\theta'|\Lambda_{\mathrm{inj}})$. This becomes difficult when training injections are themselves complex mixtures between many disparate sets of events (as in Fig. 2). By learning the latent $P(\det|\theta)$ surface itself, we avoid the downstream need to reevaluate $p(\theta|\Lambda_{\mathrm{inj}})$. Directly learning $P(\det|\theta)$ correspondingly facilitates iterative learning: if the emulator is found to perform poorly in a specific neighborhood of parameter space, additional training data can be generated in that neighborhood and the emulator retrained. This cycle can be repeated as necessary, with no ties to a globally defined $p(\theta|\Lambda_{\mathrm{inj}})$ distribution.

Instead of learning $P(\det|\theta)$, Refs. [58,59] directly emulated the integrated detection efficiency $\xi(\Lambda)$ as a function of chosen hyperparameters Λ. Reference [60] combined this idea with $P(\det|\theta)$ emulation in a two step process, training a neural network to emulate signal-to-noise ratios that were then used as training inputs for a second network emulating $\xi(\Lambda)$. The direct emulation of detection efficiencies sidesteps the (computationally burdensome) need to generate any injections and may help to avoid convergence issues associated with Monte Carlo integration. At the same time, the direct emulation of $\xi(\Lambda)$ commits oneself to a single, chosen model for the compact binary population, characterized by a specific set of hyperparameters Λ. Inference of the compact binary population with a new model would require retraining of the detection efficiency emulator, a process we wish to avoid.

As discussed above, an altogether different approach that does not rely on machine learning is to approximate search selection via a signal-to-noise ratio threshold [34,61]. As we demonstrated in Sec. V, a uniform SNR threshold applied across binary parameter space does not accurately reproduce distributions of found binary parameters, and will therefore introduce biases if used as a proxy for selection effects when analyzing gravitational-wave data. However, good performance might be achievable by calibrating a source-dependent threshold that varies across the space of binary parameters [34]. Alternatively, there exist analytic fitting functions that capture the $P(\det|\theta)$ on the lower-dimensional space of binary masses, distance, and aligned spin components [62]. Analytic methods like these have the advantage of being directly interpretable. A deep learning approach, on the other hand, trades some interpretability for speed, flexibility, and ease of generalization: a neural network like the one we present here can capture the behavior of $P(\det|\theta)$ across the high-dimensional space of compact binary parameters (including spin precession and extrinsic parameters) while requiring no direct waveform evaluation.

As continuous representations of $P(\det|\theta)$ grow in prevalence, they may influence the metrics by which pipeline injection sets are designed. Current injection sets are carefully designed to maximize the efficiencies with which they can be reweighted to other populations of interest, as in Eq. (6). When using pipeline injections as training data for $P(\det|\theta)$ representations (whether a signal-to-noise threshold, an analytic fitting function, or a neural network emulator), though, the best performance may be achieved with alternative design metrics. In the case of a neural network emulator, for example, it is beneficial to have injections uniformly placed across a much broader range of parameter space and include both very loud and very quiet events. It would be valuable for future studies to more quantitatively explore suitable metrics for the generation of training data.

Similarly, future studies should explore how emulator accuracy scales with the number of pipeline injections provided as training data. In the Advanced LIGO and

Advanced Virgo O3 observing run, 2.5×10^5 pipeline injections were performed for each of the binary black hole, binary neutron star, and neutron star-black hole populations. We used just under half of these when training our emulator (see Table II). If emulator precision could be maintained while further decreasing the number of training injections, this may minimize the future computational cost of calibrating gravitational-wave selection effects. It will also be valuable to more systematically explore different network architectures. In our study, we heuristically found that a fully-connected network with four 192-neuron wide hidden layers yielded a reasonable balance between predictive accuracy and training time. It is possible, though, that further advances (e.g. an improved loss function, alternate training data, etc.) could enable comparable accuracy with a smaller network.

Our trained detection probability emulator is made publicly available at [63]. The use of this emulator is described in Appendix A below, with more details found in documentation online.

This work made use of the following software packages: ARVIZ [64], ASTROPY [65–67], CYTHON [68], H5PY [69,70], JAX [71], MATPLOTLIB [72], NUMPY [73], NUMPYRO [74,75], PANDAS [76,77], PYTHON [78], SCIKIT-LEARN [79–81], SCIPY [82,83], and TENSORFLOW [84]. Software citation information aggregated using `The Software Citation Station` [85,86].

ACKNOWLEDGMENTS

We thank Muhammad Zeeshan, Ben Farr, Christopher Berry, and Matthew Mould for valuable comments and conversation. T. C. is supported by the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship, a Schmidt Sciences program. R. E. is supported by the Natural Sciences & Engineering Research Council of Canada (NSERC) through a Discovery Grant (No. RGPIN-2023-03346). D. E. H. is supported by NSF Grants No. AST-2006645 and No. PHY2110507, as well as by the Kavli Institute for Cosmological Physics through an endowment from the Kavli Foundation and its founder Fred Kavli. This work was completed in part with resources provided by the University of Chicago's Research Computing Center. The authors are also grateful for additional computational resources provided by the LIGO Laboratory and supported by National Science Foundation Grants No. PHY-0757058 and No. PHY-0823459. This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation.

DATA AVAILABILITY

Our trained detection probability emulator is available at [63], and code used to produce the results in this study can be found at [87]. The necessary data to regenerate figures or rerun analyses are available via Zenodo [88].

APPENDIX A: USING THE EMULATOR

A PYTHON implementation of the trained detection probability emulator is available at [63]. In this section, we briefly describe how to access and use this the trained network.

Most directly, the trained emulator can be directly imported and evaluated as illustrated in the following example:

```
from p_det import p_det_O3

# Instantiate trained emulator
p = p_det_O3()

# Define data.
# The following shows a minimal working example,
# in which we specify source-frame component
# masses, spin magnitudes, and redshifts for
# three compact binaries
params = {'mass_1':[2.5,10.0,15.0],
          'mass_2':[1.2,5.0,10.0],
          'a_1':[0.0,0.2,0.3],
          'a_2':[0.1,0.4,0.2],
          'redshift':[0.1,0.9,1.0]
         }

# Compute detection probabilities
detection_probs = p.predict(params)
```

In this example, the user has provided the minimum set of required parameters: (*i*) source-frame component masses, (*ii*) component spin magnitudes, and (*iii*) a distance parameter (either redshift, luminosity distance, or comoving distance). Additional quantities like spin orientations and extrinsic parameters can be optionally provided; if they are not provided, they are randomly generated assuming isotropy. In this example, compact binary parameters were provided in the form of a dictionary, but they may also be passed via any other structure supporting key-value functionality. Internally, the `p_det_O3.predict` method checks for the presence and self-consistency of provided parameters, transforms to the input parameter space expected by the neural network, and evaluates the network.

The above example illustrates how one might use the trained network when forward modeling sets of observable compact binary signals. As in Sec. IV, another use case is to employ the network in hierarchical inference of the compact binary population. To this end, we need an interface that is, ideally, compilable and differentiable, to enable compatibility with model likelihoods and inference performed with JAX [71] and NUMPYRO [74,75]. This is provided by calling `p_det_O3` directly (which implicitly evaluates the `p_det_O3.__call__` method) as follows:

```
from p_det import p_det_O3
import jax
import jax.numpy as jnp

# Instantiate trained emulator
p = p_det_O3()

# Obtain just-in-time-compiled probability
jitted_p_det_O3 = jax.jit(p)

# Generate and define binary parameters.
# See online documentation for the proper
# contents and formatting of this object
mass_1 = [20., 30., …]
mass_2 = [15., 29., …]
a_1 = [0.5, 0.9, …]
a_2 = [0.3, 0., …]
…
params = jnp.array([mass_1,
                    mass_2,
                    a_1,
                    a_2,
                    …])

# Compute detection probabilities
detection_probs = jitted_p_det_O3(params)
```

Direct evaluation in this manner necessarily lacks the guardrails and self-consistency checks built into the `p_det_O3.predict` method, and instead assumes that users follow a specific, expected format in providing compact binary parameters; see code documentation for exact details.

APPENDIX B: MORE ON EMULATOR TRAINING

This appendix provides additional details regarding the training data, loss function, and procedure used for neural network training.

1. Training data

As discussed in the main text, our training data comprises sets of simulated compact binaries added to Advanced LIGO and Advanced Virgo data, analyzed with search pipelines, and labeled as detected (found) or undetected (missed); see Fig. 1. Briefly, each injected population is described via a power-law primary mass distribution,

$$p(m_1|\Lambda_{\rm inj}) \propto m_1^{\alpha}(m_{1,\rm min} \leq m_1 \leq m_{1,\rm max}). \tag{B1}$$

Secondary masses are also described as a power laws, following one of two conventions. First, the secondary mass distribution can be defined conditionally on m_1, such that

$$p(m_2|m_1,\Lambda_{\rm inj}) \propto m_2^{\beta_q}\,(m_{2,\rm min} \leq m_2 \leq m_1). \tag{B2}$$

Alternatively, we can directly describe the joint distribution of m_1 and m_2 as

$$p(m_1,m_2|\Lambda_{\rm inj}) \propto m_1^{\alpha} m_2^{\beta_q}\Theta(m_1 - m_2). \tag{B3}$$

Here, $\Theta(\cdot)$ is the Heaviside step function. Note, Eq. (B3) is a different distribution than the product of Eqs. (B1) and (B2). All injection sets have independently and identically distributed component spins, with isotropic spin orientations and uniform spin magnitude distributions between $0 \leq \chi_1 \leq \chi_{1,\rm max}$ and $0 \leq \chi_2 \leq \chi_{2,\rm max}$. Volumetric merger rates evolve as a power law in $(1+z)$, such that

$$p(z|\Lambda_{\rm inj}) \propto \frac{1}{1+z}\frac{dV_c}{dz}(1+z)^{\kappa}\ \ (z \leq z_{\rm max}), \tag{B4}$$

where dV_c/dz is the differential comoving volume per unit redshift.

As described in the main text, we supplement the LIGO-Virgo-KAGRA pipeline injections with additional batches of "hopeless" events that are confidently undetectable, and "certain" events that are guaranteed to be detected if one or more LIGO instrument is in observing mode. The distributions of these hopeless and certain injection sets closely follow the LIGO-Virgo-KAGRA pipeline injections, but are chosen to have a shallower primary mass distribution (and, in the BBH case, shallower growth of the merger rate with redshift) in order to yield training data that more uniformly covers the compact binary parameter space. We additionally produced auxiliary hopeless injections spanning a broad range of masses and redshifts.

The specific hyperparameters characterizing each of these injection sets are detailed in Table II. In addition to hyperparameter values, this table also indicates which convention, Eq. (B2) or Eq. (B3), is followed when defining a secondary mass distribution. The penultimate column indicates the reference frequencies at which component spins were defined; these differ slightly between injection sets. The final column indicates the number of events drawn from each set for use as training data.

We note that the total number of available pipeline injections is larger than the numbers we used for training. As discussed in Sec. II B, we introduce additional terms in the loss function involving the integrated detection efficiencies of several reference populations (see Eq. (8) and Appendix B 2 below). These terms terms are slow to evaluate. The chosen number of injections yielded a good compromise between network accuracy and overall training time. Additionally, the specific ratios in Table II between pipeline, hopeless, and certain injections were found to yield better network performance than when simply training with additional available pipeline injections.

2. Reference populations for augmented training loss

When describing our training loss function in Sec. II B, we introduced an additional regularization term [Eq. (8)] used to motivate the network to accurately recover integrated detection efficiencies. When training our emulator, we sum Eq. (8) across four reference populations. These reference

TABLE II. Description of injection sets used to train $P(\det|\theta)$ emulator, including the hyperparameters defining each set, the convention followed when defining component mass distributions, and the number N of injections used from each set; see Sec. B 1. Note that, because $m_{2,\max} = m_{1,\min}$ for the neutron star-black hole injections, conventions (B2) and (B3) are equivalent for this population.

Injection Set	$m_{1,\min}$	$m_{1,\max}$	α	$m_{2,\min}$	$m_{2,\max}$	β_q	Convention	$\chi_{1,\max}$	$\chi_{2,\max}$	κ	$z_{\max}$	f_{ref}	N
Pipeline BBH	$2M_\odot$	$100M_\odot$	-2.35	$2M_\odot$	$100M_\odot$	1	Eq. (B2)	0.998	0.998	1	1.9	10 Hz	9×10^4
Hopeless BBH	$2M_\odot$	$100M_\odot$	-1	$2M_\odot$	$100M_\odot$	1	Eq. (B2)	0.998	0.998	0	1.9	16 Hz	1.4×10^5
Certain BBH	$2M_\odot$	$100M_\odot$	-1	$2M_\odot$	$100M_\odot$	1	Eq. (B2)	0.998	0.998	0	1.9	16 Hz	1.4×10^5
Pipeline NSBH	$2.5M_\odot$	$60M_\odot$	-2.35	$1M_\odot$	$2.5M_\odot$	0	Eq. (B2)	0.998	0.4	0	0.25	15 Hz	9×10^4
Hopeless NSBH	$2.5M_\odot$	$60M_\odot$	-1	$1M_\odot$	$2.5M_\odot$	0	Eq. (B2)	0.998	0.4	0	0.25	16 Hz	1.4×10^5
Certain NSBH	$2.5M_\odot$	$60M_\odot$	-1	$1M_\odot$	$2.5M_\odot$	0	Eq. (B2)	0.998	0.4	0	0.25	16 Hz	1.4×10^5
Pipeline BNS	$1M_\odot$	$2.5M_\odot$	0	$1M_\odot$	$2.5M_\odot$	0	Eq. (B3)	0.4	0.4	0	0.15	15 Hz	9×10^4
Hopeless BNS	$1M_\odot$	$2.5M_\odot$	0	$1M_\odot$	$2.5M_\odot$	0	Eq. (B3)	0.4	0.4	0	0.15	16 Hz	1.4×10^5
Certain BNS	$1M_\odot$	$2.5M_\odot$	0	$1M_\odot$	$2.5M_\odot$	0	Eq. (B3)	0.4	0.4	0	0.15	16 Hz	1.4×10^5
Auxiliary Hopeless	$1M_\odot$	$100M_\odot$	-2	$1M_\odot$	$100M_\odot$	-2	Eq. (B3)	0.998	0.998	-1	2	16 Hz	2.4×10^5

TABLE III. Description of the reference populations used in augmented training loss function, as defined in Eq. (8) and surrounding text. Included in the table are the hyperparameters defining each reference population, the convention used in defining a secondary mass distribution, the true detection efficiency ξ associated with each, and the number N of random draws from each population used during training.

Reference Distribution	$m_{1,\min}$	$m_{1,\max}$	α	$m_{2,\min}$	$m_{2,\max}$	β_q	Convention	$\chi_{1,\max}$	$\chi_{2,\max}$	κ	$z_{\max}$	ξ	N
BBH (“Astrophysical”)	$5M_\odot$	$100M_\odot$	-3	$2M_\odot$	$100M_\odot$	1	Eq. (B2)	0.998	0.998	4	1.9	3.8×10^{-4}	2×10^5
BBH (“Injectionlike”)	$2M_\odot$	$100M_\odot$	-2.35	$2M_\odot$	$100M_\odot$	1	Eq. (B2)	0.998	0.998	1	1.9	1.1×10^{-3}	2×10^5
NSBH (“Injectionlike”)	$2.5M_\odot$	$60M_\odot$	-2.35	$1M_\odot$	$2.5M_\odot$	0	Eq. (B3)	0.998	0.4	0	0.25	1.1×10^{-2}	10^4
BNS (“Injectionlike”)	$1M_\odot$	$2.5M_\odot$	0	$1M_\odot$	$2.5M_\odot$	0	Eq. (B3)	0.4	0.4	0	0.15	1.6×10^{-2}	10^4

populations followed the same functional forms used to define and generate training data; see Eqs. (C1)–(C7). The specific hyperparameters characterizing each are given in Table III. Also listed are the target detection efficiencies ξ_I as estimated using pipeline injections, the number N_I of draws from each population used to estimate $\hat{\xi}_I$ at each training step. The number of draws from each population were chosen to yield similar expected precisions $\sigma_\xi/\xi = 1/\sqrt{\xi_I N_I}$ for each population's detection efficiency [see Eq. (7)] while also managing local memory requirements.

3. Network structure and ensemble training

Input data are regularized via a linear transformation to the unit interval (`sklearn.StandardScaler`). The input and hidden layers use a LeakyReLU activation function with a slope parameter of 10^{-3}, while the final layer has a sigmoid activation function, rescaled to yield values on the interval $\{0, 0.94\}$. Initial neuron weights were randomly drawn from a zero-mean Gaussian distribution with standard deviation 0.01 and biases initially set to zero; the exception is the final output neuron, whose initial bias was set to $\ln(10^{-3})$.

These choices were made after experimenting with a large number of alternatives. We found that the most impactful design choices are (i) the explicit use of amplitude parameters [Eq. (9)] as well as the polarization angle as input parameters, (ii) the adoption of a relatively shallow but wide network, rather than a narrower network with more hidden layers, and (iii) the precise number of additional “certain” and “hopeless” injections we use to augment the LIGO-Virgo-KAGRA injection sets.

After finalizing all details, we trained an ensemble of approximately 50 networks, each with random initialization conditions. We graded the trained networks on two criteria. First, for each network we computed predicted distributions of compact binary parameters and computed KS-test p-values between these predictions and the distributions actually recovered via pipeline injections (as in Figs. 3–5). For each network we record the minimum p-value, taken across all three source classes and all compact binary parameters. Second, we compute predicted detection efficiencies for the reference populations listed in Table III, and, for each population, the standardized residual $(\hat{\xi} - \xi)/\sigma_\xi$ between the predicted and target values, where $\sigma_\xi = \sqrt{\xi/N}$ and N is the total number of trials performed in the given computation [see Eq. (7)]. We record the maximum standardized residual for each network. The results are shown in Fig. 13, with each point representing a trained network from among the ensemble. The fiducial network adopted for this paper is marked with a red star.

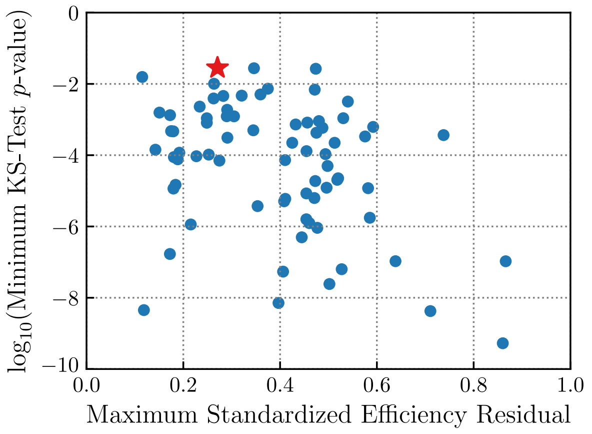

FIG. 13. Summary statistics grading trained neural networks, as described in Appendix B 3. We trained an ensemble of networks, each with a different realization of training data and random initializations. The y-axis values indicate faithfulness in recovering correct parameter distributions of found compact binaries, while x-axis values indicate network accuracy in recovering integrated detection efficiencies. The fiducial network chosen in our study is indicated with a red star.

APPENDIX C: BINARY BLACK HOLE POPULATION MODELS

In a number of places in the main text, we invoke or infer realistic models for the astrophysical population of binary black holes. This includes the discussion of integrated detection efficiencies surrounding Fig. 6 in Sec. III, and in the population inference performed in Sec. IV. In this appendix we describe the precise population models used for these results.

The primary mass distribution of binary black holes is assumed to follow a mixture between a power-law distribution and a Gaussian peak,

$$p(m_1) \propto \mathfrak{t}(m_1)\left[f_p \frac{e^{-(m_1-\mu_m)^2/2\sigma_m^2}}{\sqrt{2\pi\sigma_m^2}} + (1-f_p)\frac{(1+\alpha)m_1^{\alpha}}{(100M_\odot)^{1+\alpha}-(2M_\odot)^{1+\alpha}}\right], \quad \text{(C1)}$$

where $\mathfrak{t}(m_1)$ is a tapering function that sends the mass distribution to zero at sufficiently high and low masses:

$$\mathfrak{t}(m_1) \propto \begin{cases} e^{-(m_1-m_{\rm low})^2/2\delta m_{\rm low}^2} & (m_1 < m_{\rm low}) \\ 1 & (m_{\rm low} \leq m_1 \leq m_{\rm high}) \\ e^{-(m_1-m_{\rm high})^2/2\delta m_{\rm high}^2} & (m_1 > m_{\rm high}). \end{cases} \quad \text{(C2)}$$

Secondary masses are assumed to follow a power law, conditioned on primary masses:

$$p(m_2|m_1) = \frac{(1+\beta_q)m_2^{\beta_q}}{m_1^{1+\beta_q}-(2M_\odot)^{1+\beta_q}}. \quad \text{(C3)}$$

Component spins are assumed to be independently and identically distributed, with spin magnitudes following a truncated Gaussian distribution on the interval $0 \leq \chi \leq 1$,

$$p(\chi) = \sqrt{\frac{2}{\pi\sigma_\chi^2}}\frac{e^{-(\chi-\mu_\chi)^2/2\sigma_\chi^2}}{\mathrm{Erf}\left(\frac{1-\mu_\chi}{\sqrt{2\sigma_\chi^2}}\right)+\mathrm{Erf}\left(\frac{\mu_\chi}{\sqrt{2\sigma_\chi^2}}\right)}, \quad \text{(C4)}$$

while cosine spin-orbit tilt angles θ are assumed to follow a mixture between a truncated Gaussian and an isotropic component,

$$p(\cos\theta) = \frac{f_{\rm iso}}{2} + (1-f_{\rm iso})\sqrt{\frac{2}{\pi\sigma_u^2}}\frac{e^{-(\cos\theta-\mu_u)^2/2\sigma_u^2}}{\mathrm{Erf}\left(\frac{1-\mu_u}{\sqrt{2\sigma_u^2}}\right)+\mathrm{Erf}\left(\frac{1+\mu_u}{\sqrt{2\sigma_u^2}}\right)}, \quad \text{(C5)}$$

defined on the interval $-1 \leq \cos\theta \leq 1$. The black hole merger rate per unit volume is assumed to evolve as a power law in $1+z$, such that the total number of mergers per unit source-frame time dt_s per unit comoving volume dV_c is

$$\frac{dN}{dt_s dV_c}(z) \propto (1+z)^\kappa. \quad \text{(C6)}$$

The corresponding probability distribution of binary black hole redshifts is

$$p(z) \propto \frac{dV_c}{dz}(1+z)^{\kappa-1}, \quad \text{(C7)}$$

where dV_c/dz is the differential comoving volume per unit redshift and the additional factor of $1+z$ converts from source-frame to detector-frame time.

Table IV gives the priors and/or values adopted for the hyperparameters describing Eqs. (C1)–(C7) at different points throughout the paper. The second column shows the hyperparameter distributions sampled to obtain the results in Fig. 6. The third column gives the fixed values chosen for Fig. 7 (the value of σ_χ is left blank, as this parameter is varied) when demonstrating the dynamic regeneration of injections using the $P(\det|\theta)$ emulator. And the final column gives the hyperpriors adopted when performing full hierarchical inference in Sec. IV, both when using traditional injection reweighting and when instead leveraging the neural network emulator.

APPENDIX D: HIERARCHICAL INFERENCE METHODS AND HYBRID INJECTION GENERATION

We perform hierarchical inference of the binary black hole population following standard methods, as described in, e.g., Refs. [27,28]. Consider a set of detected gravitational-wave events, with data $\{d_I\}$, where $I \in [1, N_{\rm events}]$ indexes each event. We have posterior samples $\{\theta_{I,j}\}$ on the properties of each event, generated according to some generic prior $p(\theta|\Lambda_{\rm pe})$. The

TABLE IV. Hyperparameters specifying the binary black hole population models used throughout this work, as defined in Appendix C. The second column defines the distributions randomly sampled to obtain Fig. 6. The third column gives the fixed values adopted when producing Fig. 7 (σ_χ is varied in this figure, and so is not given a value below). Finally, the fourth column gives the priors adopted when hierarchically inferring the binary black hole population in Sec. IV and Fig. 8.

Parameter	Detection Efficiencies (Fig. 6)	Dynamic injection regeneration (Fig. 7)	Hierarchical Inference (Fig. 8)
μ_m	$\mathrm{U}(20M_\odot, 50M_\odot)$	$35M_\odot$	$\mathrm{U}(20M_\odot, 50M_\odot)$
σ_m	$\mathrm{U}(2M_\odot, 15M_\odot)$	$5M_\odot$	$\mathrm{U}(2M_\odot, 15M_\odot)$
f_p	$\mathrm{LU}(10^{-6}, 1)$	10^{-3}	$\mathrm{LU}(10^{-6}, 1)$
α	$\mathrm{N}(-2, 3)$	-3	$\mathrm{N}(-2, 3)$
m_{low}	$\mathrm{U}(5M_\odot, 15M_\odot)$	$10M_\odot$	$\mathrm{U}(5M_\odot, 15M_\odot)$
δm_{low}	$\mathrm{LU}(0.1M_\odot, 10M_\odot)$	$1M_\odot$	$\mathrm{LU}(0.1M_\odot, 10M_\odot)$
m_{high}	$\mathrm{U}(50M_\odot, 100M_\odot)$	$80M_\odot$	$\mathrm{U}(50M_\odot, 100M_\odot)$
δm_{high}	$\mathrm{LU}(10^{0.5}M_\odot, 10^{1.5}M_\odot)$	$10M_\odot$	$\mathrm{LU}(10^{0.5}M_\odot, 10^{1.5}M_\odot)$
β_q	$\mathrm{N}(0, 3)$	2	$\mathrm{N}(0, 3)$
μ_χ	$\mathrm{U}(0, 1)$	0	$\mathrm{U}(0, 1)$
σ_χ	$\mathrm{LU}(0.1, 1)$	…	$\mathrm{LU}(0.1, 1)$
f_{iso}	$\mathrm{U}(0, 1)$	0.5	$\mathrm{U}(0, 1)$
μ_u	$\mathrm{U}(-1, 1)$	1	$\mathrm{U}(-1, 1)$
σ_u	$\mathrm{U}(0.15, 2.5)$	0.5	$\mathrm{U}(0.15, 2.5)$
κ	$\mathrm{N}(0, 5)$	3	$\mathrm{N}(0, 5)$

likelihood that this data arose from a given compact binary population Λ is, then,

$$p(\{d\}|\Lambda) \propto e^{-N_{\mathrm{exp}}(\Lambda)} \prod_{I=1}^{N_{\mathrm{events}}} \left\langle \frac{dN/d\theta(\theta_{I,j};\Lambda)}{p(\theta_{I,j}|\Lambda_{\mathrm{pe}})} \right\rangle_j, \tag{D1}$$

where $\langle\cdot\rangle_j$ denotes an ensemble average over the posterior samples j.

The factor $dN/d\theta(\theta;\Lambda)$ in Eq. (D1) is the predicted number density of compact binary mergers (e.g. number of events per unit redshift, per unit mass, etc.). This is related to the volumetric source-frame rate by

$$\frac{dN}{d\theta} = \frac{T_{\mathrm{obs}}}{1+z}\frac{dV_c}{dz}\frac{dN}{dt_s dV_c dm_1 dq\ldots}, \tag{D2}$$

where T_{obs} is our experiment's total duration. We define the source-frame rate following the models described in Appendix C. In particular, it is convenient to parametrize this function as

$$\frac{dN}{dt_s dV_c dm_1 dq\ldots} = R_{\mathrm{ref}}\left(\frac{1+z}{1+z_{\mathrm{ref}}}\right)^{\kappa}\frac{p(m_1)}{p(m_1 = m_{\mathrm{ref}})} \times p(q)p(\chi_1)p(\chi_2)p(\cos\theta_1)p(\cos\theta_2), \tag{D3}$$

which avoids numerical computation of the normalization coefficient of the mass distribution. This implies that R_{ref} is defined to be the volumetric merger rate per unit mass, defined at $m_1 = m_{\mathrm{ref}}$ and $z = z_{\mathrm{ref}}$; we take $m_{\mathrm{ref}} = 20M_\odot$ and $z = 0.2$.

The term $N_{\mathrm{exp}}(\Lambda)$, meanwhile, is the expected number of detections if the compact binary population were indeed described by Λ. This is directly related to the detection efficiency, $N_{\mathrm{exp}}(\Lambda) = N(\Lambda)\xi(\Lambda)$, where $N(\Lambda)$ is the total number of events occurring in the observation period, as can be obtained by integrating $dN/d\theta$. When correcting for selection effects using a fixed suite of pipeline injections, $N_{\mathrm{exp}}(\Lambda)$ can be more directly estimated via

$$N_{\mathrm{exp}}(\Lambda) = \frac{1}{N_{\mathrm{total}}}\sum_i \frac{dN/d\theta(\theta_i;\Lambda)}{p(\theta_i|\Lambda_{\mathrm{inj}})}, \tag{D4}$$

where the index i ranges over found injections, N_{total} is the total number of injections performed, and $p(\theta|\Lambda_{\mathrm{inj}})$ is the distribution from which the injections were drawn; compare to Eq. (6). As described in the main text, we can alternatively use a trained $P(\det|\theta)$ emulator to evaluate N_{exp}. Following the algorithm in Sec. IV A, we directly generate a set of compact binary parameters $\{\theta_i\} \sim p(\theta|\Lambda)$ drawn from the proposed population Λ. Then the expected number of detections is

$$\begin{aligned} N_{\mathrm{exp}}(\Lambda) &= \int \frac{dN}{d\theta}(\theta;\Lambda)P(\det|\theta)d\theta \\ &= N(\Lambda)\int p(\theta|\Lambda)P(\det|\theta)d\theta \\ &\approx N(\Lambda)\langle\hat{P}(\det|\theta_i)\rangle_i, \end{aligned} \tag{D5}$$

replacing the integral of $P(\det|\theta)$ over the compact binary probability distribution with the ensemble average of our emulator, $\hat{P}(\det|\theta)$, across the samples drawn from $p(\theta|\Lambda)$. The remaining term $N(\Lambda)$ is the expected total number of events (detected or otherwise); this is found by integrating Eq. (D2). Plugging in our model for the source-frame rate density [Eq. (D3)], this yields

$$N(\Lambda) = R_{\text{ref}} \frac{\int p(m_1) dm_1}{p(m_1 = m_{\text{ref}})} \frac{\int dV_c/dz (1+z)^{\kappa-1} dz}{(1+z_{\text{ref}})^{\kappa}}. \tag{D6}$$

In practice, we find that the above approach does not always yield stable results. Specifically, while the large majority of samples drawn from realistic populations will lie at high redshifts, the detection efficiency primarily depends on the small fraction that happen to lie at low redshift, yielding a high-variance estimate of the integrated detection efficiency. To circumvent this, we modify the algorithm in Sec. IV A by, in Step 1, drawing a single, *fixed* set of redshifts $z \sim p(z|\kappa = \kappa_{\text{ref}})$ from a reference redshift distribution defined by some κ_{ref}. We choose $\kappa_{\text{ref}} = -1.5$ in order to guarantee a large number of injections situated at small redshifts. All other parameters are dynamically drawn as usual following the algorithm in Sec. IV A. Evaluation of

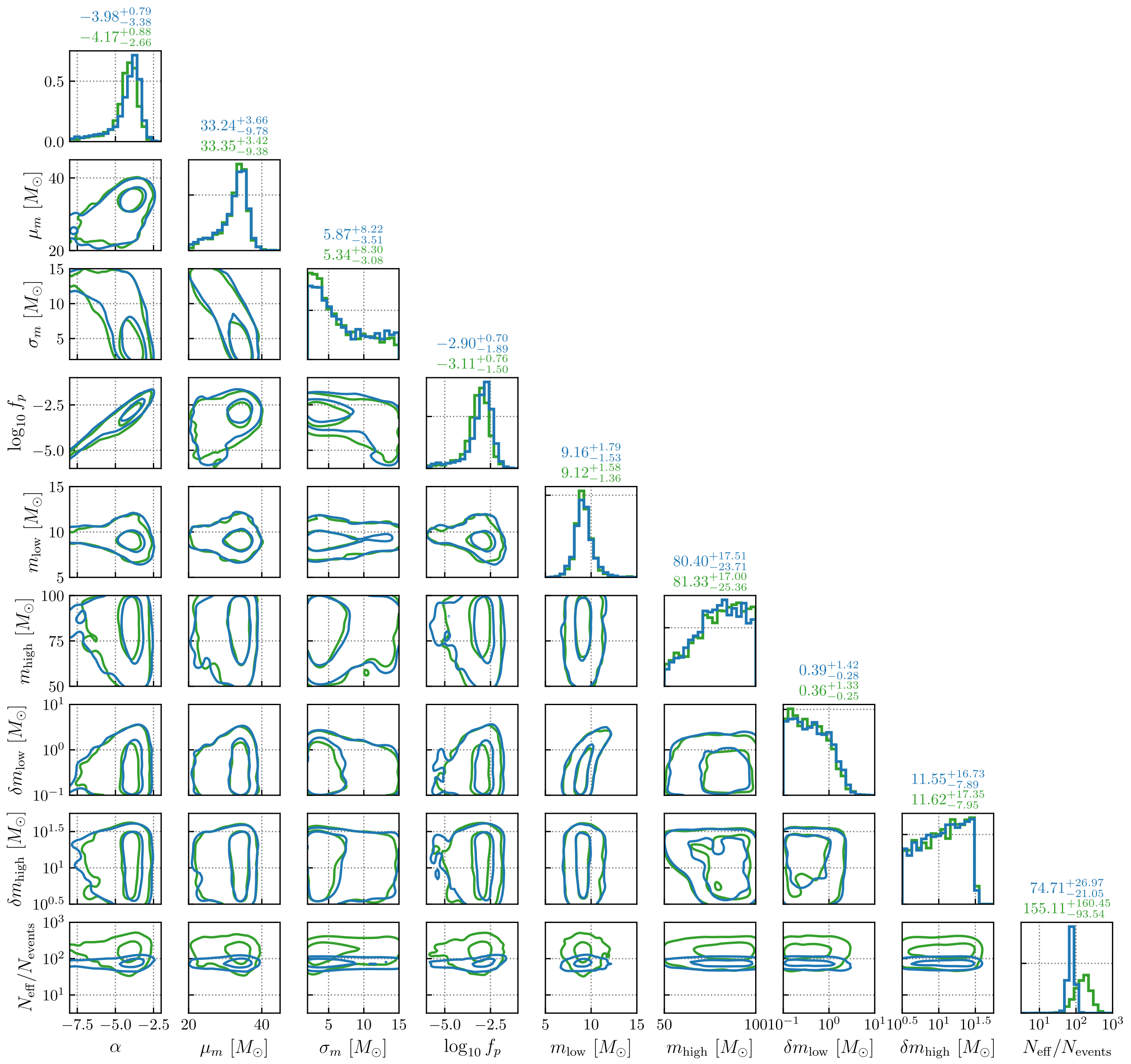

FIG. 14. Inferred posteriors on the hyperparameters governing the binary black hole primary mass distribution, as discussed in Sec. IV. Results using standard reweighting of pipeline injections are shown in green, while results obtained using our neural network $P(\det|\theta)$ emulator are given in blue. The labels above each column give the median inferred hyperparameter values, with errors indicating central 95% credible bounds.

$N_{\rm exp}(\Lambda)$ then proceeds as above, but with an extra reweighting term accompanying the ensemble average over injections. Define $\tilde{\theta}$ as the set of all binary parameters *except* redshift, and similarly $\tilde{\Lambda}$ to be all population hyperparameters except κ. Then

$$\begin{aligned} N_{\rm exp}(\Lambda) &= N(\Lambda)\int p(\tilde{\theta}|\tilde{\Lambda})p(z|\kappa)P(\det|\tilde{\theta},z)d\tilde{\theta}dz \\ &= N(\Lambda)\int p(\tilde{\theta}|\tilde{\Lambda})p(z|\kappa_{\rm ref})\frac{p(z|\kappa)}{p(z|\kappa_{\rm ref})} \\ &\quad\times P(\det|\tilde{\theta},z)d\tilde{\theta}dz \\ &\approx N(\Lambda)\left\langle\frac{p(z_i|\kappa)}{p(z_i|\kappa_{\rm ref})}\hat{P}(\det|\tilde{\theta}_i,z_i)\right\rangle_i, \end{aligned} \tag{D7}$$

where in the final line we are now taking the average over our hybrid injections drawn from the fixed redshift distribution $p(z|\kappa_{\rm ref})$. Note that the probability distributions appearing in the reweighting factors must be properly normalized [although the normalization coefficient of $p(z_i|\kappa)$ conveniently cancels with the redshift integral appearing in Eq. (D6) for $N(\Lambda)$].

We perform inference using the "No U-Turn" sampler [89] implemented in NUMPYRO [74,75], a probabilistic programming library built atop JAX [71]. Our priors are as described in Appendix C and Table IV therein. We use posterior samples from the GWTC-2.1 [54] and GWTC-3 catalogs [42], made available via Zenodo [90,91]. We specifically make use of the "`C01:Mixed`" samples, comprising results from a mixture of waveform models that all include the effects of spin precession and radiation from higher-order multipole moments. For all inference runs, we monitor convergence by tracking the effective number of injections per catalog event [see Eq. (13)], together with the effective number of posterior samples informing the likelihood assigned to each event.

APPENDIX E: ADDITIONAL RESULTS

In the main text, we showed hierarchical inference results in the form of direct constraints on the distributions of black holes masses, spins, and redshifts. In Figs. 14–16, we alternatively show posteriors on the hyperparameters

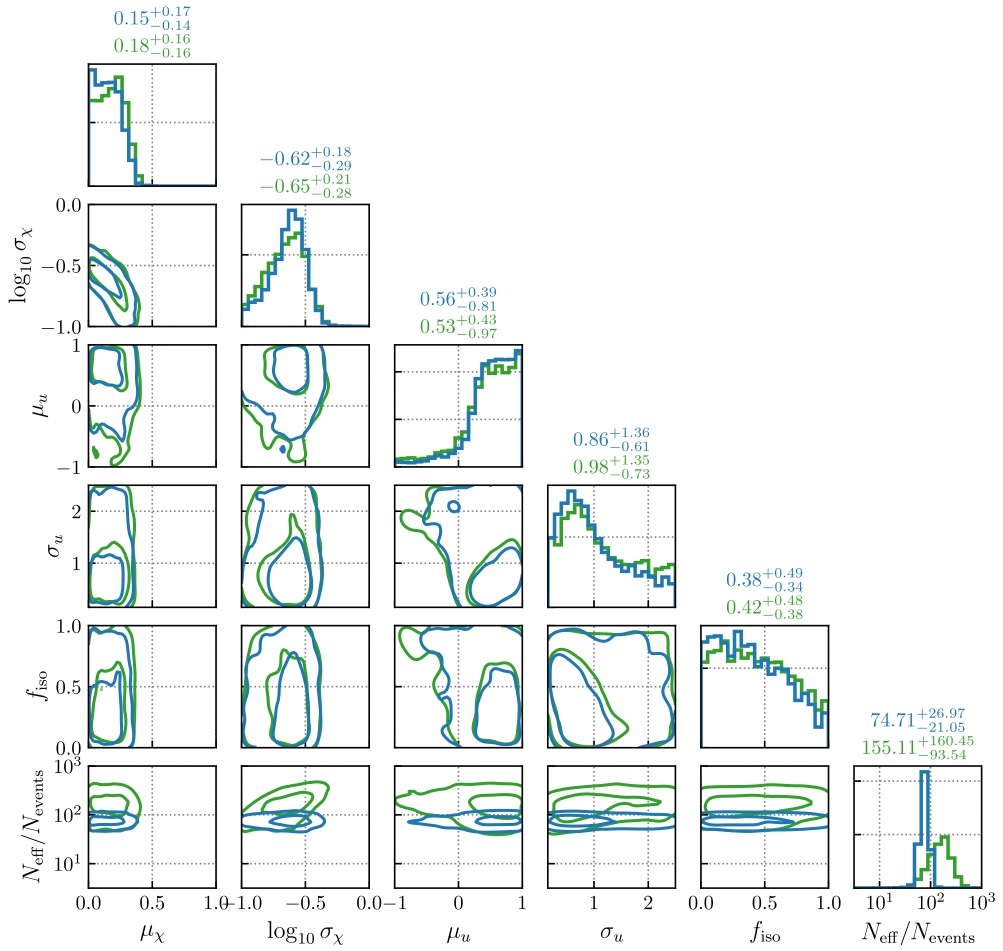

FIG. 15. As in Fig. 14, but showing hyperparameters associated with the binary black hole component spin distribution.

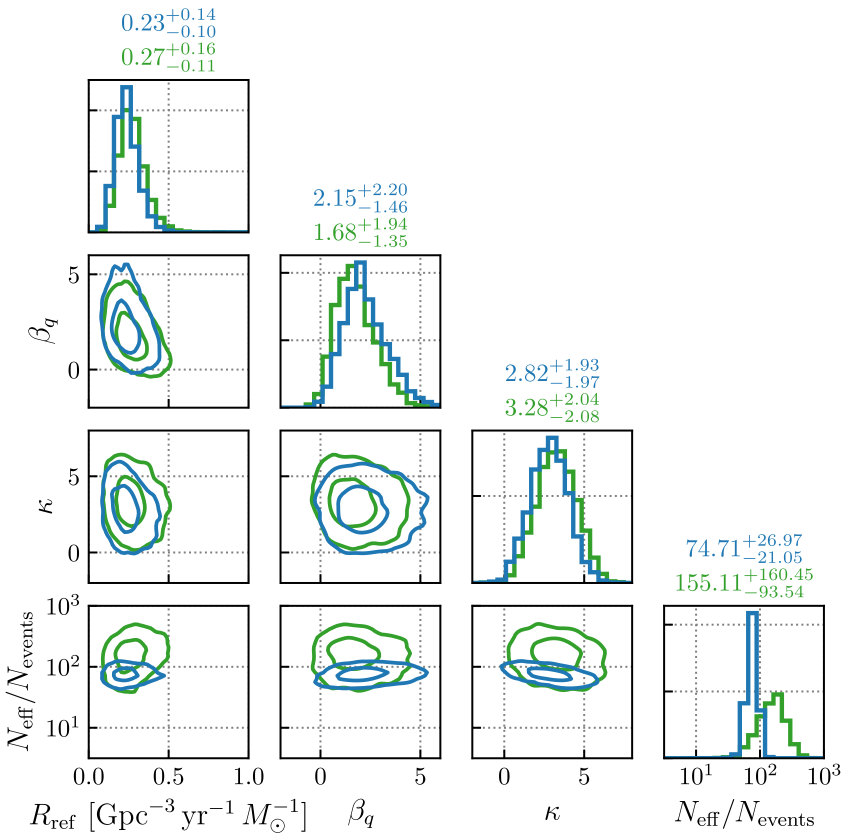

FIG. 16. As in Fig. 14, but showing hyperparameters governing the mass ratio distribution and redshift-dependent merger rate of binary black holes. Among all the hyperparameters across Figs. 14–16, β_q and κ [see Eqs. (C3) and (C6)] show the largest differences between both inference methods. When inferring the binary black hole population using our $P(\det|\theta)$ emulator, we recover β_q and κ posteriors shifted towards larger and smaller values, respectively, relative to posteriors obtained through fixed injection reweighting.

underlying these distributions. As in the main text, green curves show posteriors obtained via traditional injection reweighting, while blue curves show results obtained using our neural network $P(\det|\theta)$ emulator. Both sets of results are broadly consistent with on another, with the largest differences being slight shifts in the β_q and κ posteriors in Fig. 16. In each corner plot, we also show the effective number of injections (whether fixed injections used for reweighting or freshly generated using the neural network) per event in the catalog sample, $N_{\rm eff}/N_{\rm events}$. Hierarchical inference with injection reweighting yields a large range of effective injection counts, with values that are correlated with certain hyperparameters (e.g. α, σ_m, and σ_u). In contrast, using a $P(\det|\theta)$ emulator to draw from each target population of interest yields $N_{\rm eff}/N_{\rm events}$ values that are largely consistent across the full posterior and uncorrelated with specific hyperparameters. The one exception is κ, which is slightly correlated with effective injection counts due to our use of the hybrid injection generation algorithm described above.

[1] J. Aasi, B. P. Abbott, R. Abbott, T. Abbott, M. R. Abernathy *et al.* (The LIGO Scientific Collaboration), Advanced LIGO, Classical Quantum Gravity **32**, 074001 (2015).
[2] F. Acernese, M. Agathos, K. Agatsuma, D. Aisa, N. Allemandou *et al.* (The Virgo Collaboration), Advanced Virgo: A second-generation interferometric gravitational wave detector, Classical Quantum Gravity **32**, 024001 (2015).
[3] B. F. Schutz and M. Tinto, Antenna patterns of interferometric detectors of gravitational waves—I. Linearly polarized waves, Mon. Not. R. Astron. Soc. **224**, 131 (1987).

[4] L. S. Finn and D. F. Chernoff, Observing binary inspiral in gravitational radiation: One interferometer, Phys. Rev. D **47**, 2198 (1993).

[5] C. Cutler and E. E. Flanagan, Gravitational waves from merging compact binaries: How accurately can one extract the binary's parameters from the inspiral waveform?, Phys. Rev. D **49**, 2658 (1994).

[6] E. E. Flanagan and S. A. Hughes, Measuring gravitational waves from binary black hole coalescences. I. Signal to noise for inspiral, merger, and ringdown, Phys. Rev. D **57**, 4535. (1998).

[7] T. Dal Canton, A. H. Nitz, A. P. Lundgren, A. B. Nielsen, D. A. Brown *et al.*, Implementing a search for aligned-spin neutron star-black hole systems with advanced ground based gravitational wave detectors, Phys. Rev. D **90**, 082004 (2014).

[8] T. Adams, D. Buskulic, V. Germain, G. M. Guidi, F. Marion *et al.*, Low-latency analysis pipeline for compact binary coalescences in the advanced gravitational wave detector era, Classical Quantum Gravity **33**, 175012 (2016).

[9] S. Klimenko, G. Vedovato, M. Drago, F. Salemi, V. Tiwari *et al.*, Method for detection and reconstruction of gravitational wave transients with networks of advanced detectors, Phys. Rev. D **93**, 042004 (2016).

[10] S. A. Usman, A. H. Nitz, I. W. Harry, C. M. Biwer, D. A. Brown *et al.*, The PyCBC search for gravitational waves from compact binary coalescence, Classical Quantum Gravity **33**, 215004 (2016).

[11] C. Messick, K. Blackburn, P. Brady, P. Brockill, K. Cannon *et al.*, Analysis framework for the prompt discovery of compact binary mergers in gravitational-wave data, Phys. Rev. D **95**, 042001 (2017).

[12] S. Sachdev, S. Caudill, H. Fong, R. K. L. Lo, C. Messick *et al.*, The GstLAL search analysis methods for compact binary mergers in advanced LIGO's second and Advanced Virgo's first observing runs, arXiv:1901.08580.

[13] F. Aubin, F. Brighenti, R. Chierici, D. Estevez, G. Greco *et al.*, The MBTA pipeline for detecting compact binary coalescences in the third LIGO–Virgo observing run, Classical Quantum Gravity **38**, 095004 (2021).

[14] M. Drago *et al.*, Coherent WaveBurst, A pipeline for unmodeled gravitational-wave data analysis, SoftwareX **14**, 100678 (2021).

[15] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese *et al.*, The rate of binary black hole mergers inferred from Advanced LIGO observations surrounding GW150914, Astrophys. J. **833**, L1 (2016).

[16] C. Biwer, D. Barker, J. Batch, J. Betzwieser, R. Fisher *et al.*, Validating gravitational-wave detections: The Advanced LIGO hardware injection system, Phys. Rev. D **95**, 062002 (2017).

[17] B. P. Abbott, R. Abbott, T. D. Abbott, S. Abraham, F. Acernese *et al.* (The LIGO Scientific and the Virgo Collaborations), Binary black hole population properties inferred from the first and second observing Runs of Advanced LIGO and Advanced Virgo, Astrophys. J. **882**, L24 (2019).

[18] R. Abbott, T. D. Abbott, S. Abraham, F. Acernese, K. Ackley *et al.* (The LIGO Scientific and the Virgo Collaborations), Population properties of compact objects from the second LIGO–Virgo gravitational-wave transient catalog, Astrophys. J. Lett. **913**, L7 (2021).

[19] R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams *et al.* (The LIGO Scientific, Virgo, and KAGRA Collaborations), The population of merging compact binaries inferred using gravitational waves through GWTC-3, Phys. Rev. X **13**, 011048 (2023).

[20] M. Dominik, E. Berti, R. O'Shaughnessy, I. Mandel, K. Belczynski *et al.*, Double compact objects. III. gravitational-wave detection rates, Astrophys. J. **806**, 263 (2015).

[21] C. L. Rodriguez, M. Zevin, P. Amaro-Seoane, S. Chatterjee, K. Kremer *et al.*, Black holes: The next generation—repeated mergers in dense star clusters and their gravitational-wave properties, Phys. Rev. D **100**, 043027 (2019).

[22] K. Belczynski, J. Klencki, C. E. Fields, A. Olejak, E. Berti *et al.*, Evolutionary roads leading to low effective spins, high black hole masses, and O1/O2 rates for LIGO/Virgo binary black holes, Astron. Astrophys. **636**, 40 (2020).

[23] S. S. Bavera, M. Fishbach, M. Zevin, E. Zapartas, and T. Fragos, The $\chi_{\mathrm{eff}} - z$ correlation of field binary black hole mergers and how 3G gravitational-wave detectors can constrain it, Astron. Astrophys. **665**, A59 (2022).

[24] F. S. Broekgaarden, E. Berger, S. Stevenson, S. Justham, I. Mandel *et al.*, Impact of massive binary star and cosmic evolution on gravitational wave observations—II. Double compact object rates and properties, Mon. Not. R. Astron. Soc. **516**, 5737 (2022).

[25] G. Fragione and F. A. Rasio, Demographics of hierarchical black hole mergers in dense star clusters, Astrophys. J. **951**, 129 (2023).

[26] W. M. Farr, J. R. Gair, I. Mandel, and C. Cutler, Counting and confusion: Bayesian rate estimation with multiple populations, Phys. Rev. D **91**, 023005 (2015).

[27] I. Mandel, W. M. Farr, and J. R. Gair, Extracting distribution parameters from multiple uncertain observations with selection biases, Mon. Not. R. Astron. Soc. **486**, 1086 (2019).

[28] S. Vitale, D. Gerosa, W. M. Farr, and S. R. Taylor, Inferring the properties of a population of compact binaries in presence of selection effects, in *Handbook of Gravitational Wave Astronomy* (Springer, Singapore, 2020), 10.1007/978-981-15-4702-7_45-1.

[29] V. Tiwari, Estimation of the sensitive volume for gravitational-wave source populations using weighted Monte Carlo integration, Classical Quantum Gravity **35**, 145009 (2018).

[30] W. M. Farr, Accuracy requirements for empirically-measured selection functions, Res. Notes Am. Astron. Soc. **3**, 66 (2019).

[31] R. Essick, Constructing mixture models for sensitivity estimates from subsets of separate injections, Res. Notes Am. Astron. Soc. **5**, 220 (2021).

[32] R. Essick and W. Farr, Precision requirements for Monte Carlo sums within hierarchical Bayesian inference, arXiv:2204.00461.

[33] The LIGO Scientific, Virgo, and KAGRA Collaborations, GWTC-3: Compact binary coalescences observed by LIGO and Virgo during the second part of the third observing run—O1 + O2 + O3 search sensitivity estimates, 10.5281/zenodo.5636816 (2021).

[34] R. Essick, Semianalytic sensitivity estimates for catalogs of gravitational-wave transients, Phys. Rev. D **108**, 043011 (2023).

[35] C. Talbot and J. Golomb, Growing pains: Understanding the impact of likelihood uncertainty on hierarchical Bayesian inference for gravitational-wave astronomy, Mon. Not. R. Astron. Soc. **526**, 3495 (2023).

[36] M. Fishbach, D. E. Holz, and W. M. Farr, Does the black hole merger rate evolve with redshift?, Astrophys. J. **863**, L41 (2018).

[37] M. Fishbach, W. M. Farr, and D. E. Holz, The most massive binary black hole detections and the identification of population outliers, Astrophys. J. Lett. **891**, L31 (2020).

[38] D. Gerosa, G. Pratten, and A. Vecchio, Gravitational-wave selection effects using neural-network classifiers, Phys. Rev. D **102**, 103020 (2020).

[39] A. Farah, M. Fishbach, B. Edelman, M. Zevin, and J. M. Ezquiaga, gwmockcat, https://git.ligo.org/amanda.farah/GWMockCat.

[40] B. Allen, χ^2 time-frequency discriminator for gravitational wave detection, Phys. Rev. D **71**, 062001 (2005).

[41] R. Abbott, H. Abe, F. Acernese, K. Ackley, S. Adhicary *et al.*, Open data from the third observing run of LIGO, Virgo, KAGRA, and GEO, Astrophys. J. Suppl. Ser. **267**, 29 (2023).

[42] R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams *et al.* (The LIGO Scientific, Virgo, and KAGRA Collaborations), GWTC-3: Compact binary coalescences observed by LIGO and Virgo during the second part of the third observing run, Phys. Rev. X **13**, 041039 (2023).

[43] A. Buonanno, B. R. Iyer, E. Ochsner, Y. Pan, and B. S. Sathyaprakash, Comparison of post-Newtonian templates for compact binary inspiral signals in gravitational-wave detectors, Phys. Rev. D **80**, 084043 (2009).

[44] E. Racine, Analysis of spin precession in binary black hole systems including quadrupole-monopole interaction, Phys. Rev. D **78**, 044021 (2008).

[45] L. Santamaría, F. Ohme, P. Ajith, B. Brügmann, N. Dorband *et al.*, Matching post-Newtonian and numerical relativity waveforms: Systematic errors and a new phenomenological model for nonprecessing black hole binaries, Phys. Rev. D **82**, 064016 (2010).

[46] P. Ajith, M. Hannam, S. Husa, Y. Chen, B. Brügmann *et al.*, Inspiral-merger-ringdown waveforms for black-hole binaries with nonprecessing spins, Phys. Rev. Lett. **106**, 241101 (2011).

[47] L. E. Kidder, C. M. Will, and A. G. Wiseman, Spin effects in the inspiral of coalescing compact binaries, Phys. Rev. D **47**, R4183 (1993).

[48] D. Gerosa, M. Mould, D. Gangardt, P. Schmidt, G. Pratten, and L. M. Thomas, A generalized precession parameter χ_p to interpret gravitational-wave data, Phys. Rev. D **103**, 064067 (2021).

[49] P. Schmidt, F. Ohme, and M. Hannam, Towards models of gravitational waveforms from generic binaries: II. Modelling precession effects with a single effective precession parameter, Phys. Rev. D **91**, 024043 (2015).

[50] W. Farr and B. Farr (private Communication).

[51] H. Niederreiter, Low-discrepancy and low-dispersion sequences, J. Number Theory **30**, 51 (1988).

[52] T. A. Callister, S. J. Miller, K. Chatziioannou, and W. M. Farr, No evidence that the majority of black holes in binaries have zero spin, Astrophys. J. Lett. **937**, L13 (2022).

[53] R. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), GW190814: Gravitational waves from the coalescence of a 23 solar mass black hole with a 2.6 solar mass compact object, Astrophys. J. **896**, L44 (2020).

[54] R. Abbott, T. Abbott, F. Acernese, K. Ackley, and C. A. Adams, GWTC-2.1: Deep extended catalog of compact binary coalescences observed by LIGO and Virgo during the first half of the third observing run, Phys. Rev. D **109**, 022001 (2024).

[55] C. Talbot and E. Thrane, Measuring the binary black hole mass spectrum with an astrophysically motivated parameterization, Astrophys. J. **856**, 173 (2018).

[56] M. Fishbach and D. E. Holz, Picky partners: The pairing of component masses in binary black hole mergers, Astrophys. J. Lett. **891**, L27 (2020).

[57] C. Talbot and E. Thrane, Flexible and accurate evaluation of gravitational-wave malmquist bias with machine learning, Astrophys. J. **927**, 76 (2022).

[58] K. W. Wong, K. Breivik, K. Kremer, and T. Callister, Joint constraints on the field-cluster mixing fraction, common envelope efficiency, and globular cluster radii from a population of binary hole mergers via deep learning, Phys. Rev. D **103**, 083021 (2021).

[59] M. Mould, D. Gerosa, and S. R. Taylor, Deep learning and Bayesian inference of gravitational-wave populations: Hierarchical black-hole mergers, Phys. Rev. D **106**, 103013 (2022).

[60] C. E. A. Chapman-Bird, C. P. L. Berry, and G. Woan, Rapid determination of *LISA* sensitivity to extreme mass ratio inspirals with machine learning, Mon. Not. R. Astron. Soc. **522**, 6043 (2023).

[61] D. Gerosa and M. Bellotti, Quick recipes for gravitational-wave selection effects, Classical Quantum Gravity **41**, 125002 (2024).

[62] A. Lorenzo-Medina and T. Dent, A physically modelled selection function for compact binary mergers in the LIGO-Virgo O3 run and beyond, arXiv:2408.13383.

[63] T. Callister, R. Essick, and D. Holz, `pdet` https://github.com/tcallister/pdet (2024).

[64] R. Kumar, C. Carroll, A. Hartikainen, and O. Martin, ARVIZ a unified library for exploratory analysis of bayesian models in PYTHON, J. Open Source Software **4**, 1143 (2019).

[65] T. P. Robitaille, E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray *et al.* (ASTROPY Collaboration), ASTROPY: A community PYTHON package for astronomy, Astron. Astrophys. **558**, A33 (2013).

[66] ASTROPY Collaboration, A. M. Price-Whelan, B. M. Sipőcz, H. M. Günther, P. L. Lim, S. M. Crawford *et al.*, The ASTROPY project: Building an open-science project and status of the v2.0 core package, Astron. J. **156**, 123 (2018).

[67] Astropy Collaboration, A. M. Price-Whelan, P. L. Lim, N. Earl, N. Starkman, L. Bradley *et al.*, The ASTROPY project: Sustaining and growing a community-oriented open-source project and the latest major release (v5.0) of the core package, Astrophys. J. **935**, 167 (2022).

[68] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, CYTHON: The best of both worlds, Comput. Sci. Eng. **13**, 31 (2011).

[69] A. Collette, *Python and HDF5* (O'Reilly, 2013).

[70] A. Collette, T. Kluyver, T. A. Caswell, J. Tocknell, J. Kieffer *et al.*, h5py/h5py: 3.8.0 (2023), https://zenodo.org/records/7560547.

[71] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary *et al.*, JAX: Composable transformations of Python+NumPy programs (2018), https://github.com/jax-ml/jax.

[72] J. D. Hunter, MATPLOTLIB: A 2d graphics environment, Comput. Sci. Eng. **9**, 90 (2007).

[73] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen *et al.*, Array programming with NUMPY, Nature (London) **585**, 357 (2020).

[74] D. Phan, N. Pradhan, and M. Jankowiak, Composable effects for flexible and accelerated probabilistic programming in NUMPYRO, arXiv:1912.11554.

[75] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan *et al.*, Pyro: Deep universal probabilistic programming, J. Mach. Learn. Res. **20**, 1 (2019), https://jmlr.org/papers/v20/18-403.html.

[76] Wes McKinney, Data structures for statistical computing in PYTHON, in *Proceedings of the 9th PYTHON in Science Conference*, edited by Stéfan van der Walt and Jarrod Millman (2010), pp. 56–61, https://proceedings.scipy.org/articles/Majora-92bf1922-00a.

[77] Pandas Development Team, pandas-dev/pandas: Pandas (2024), https://zenodo.org/records/10957263.

[78] G. Van Rossum and F. L. Drake, *PYTHON3 Reference Manual* (CreateSpace, Scotts Valley, CA, 2009).

[79] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion *et al.*, SCIKIT-LEARN: Machine learning in PYTHON, J. Mach. Learn. Res. **12**, 2825 (2011), https://www.jmlr.org/papers/v12/pedregosa11a.html.

[80] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller *et al.*, API design for machine learning software: Experiences from the SCIKIT-LEARN project, in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013), pp. 108–122.

[81] O. Grisel, A. Mueller, Lars, A. Gramfort, G. Louppe *et al.*, scikit-learn/scikit-learn: SCIKIT-LEARN 1.5.0 (2024), https://zenodo.org/records/11237090.

[82] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy *et al.*, SCIPY1.0: Fundamental algorithms for scientific computing in PYTHON, Nat. Methods **17**, 261 (2020).

[83] R. Gommers, P. Virtanen, M. Haberland, E. Burovski, W. Weckesser *et al.*, scipy/scipy: SCIPY1.13.1 (2024), https://zenodo.org/records/11255513.

[84] T. Developers, TENSORFLOW, 10.5281/zenodo.12119782 (2024).

[85] T. Wagg and F. S. Broekgaarden, Streamlining and standardizing software citations with the software citation station, arXiv:2406.04405.

[86] T. Wagg, F. Broekgaarden, and K. Gültekin, Tomwagg/software-citation-station: V1.2 (2024), https://zenodo.org/records/13225824.

[87] T. Callister, R. Essick, and D. Holz, `learning-p-det` https://github.com/tcallister/learning-p-det (2024).

[88] T. Callister, R. Essick, and D. Holz, Data release: "A neural network emulator of the Advanced LIGO and Advanced Virgo selection function" (2024), 10.5281/zenodo.13362691.

[89] M. D. Hoffman and A. Gelman, The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo, J. Mach. Learn. Res. **15**, 1593 (2014), https://jmlr.org/papers/v15/hoffman14a.html.

[90] LIGO Scientific and Virgo Collaborations, GWTC-2.1: Deep extended catalog of compact binary coalescences observed by LIGO and Virgo during the first half of the third observing run—parameter estimation data release, 10.5281/zenodo.6513631 (2022).

[91] LIGO Scientific, Virgo, and KAGRA Collaborations, GWTC-3: Compact binary coalescences observed by LIGO and Virgo during the second part of the third observing run—parameter estimation data release, 10.5281/zenodo.8177023 (2023).