





Surrogate Forward Models for Population Inference on Compact Binary Mergers

Jeff Riley^{1,2}  and Ilya Mandel^{1,2} ¹ School of Physics and Astronomy, Monash University, Clayton, Victoria 3800, Australia; jeff.riley@monash.edu² ARC Centre of Excellence for Gravitational Wave Discovery—OzGrav, Australia

Received 2023 February 28; revised 2023 April 6; accepted 2023 April 21; published 2023 June 13

Abstract

Rapidly growing catalogs of compact binary mergers from advanced gravitational wave detectors allow us to explore the astrophysics of massive stellar binaries. Merger observations can constrain the uncertain parameters that describe the underlying processes in the evolution of stars and binary systems in population models. In this paper, we demonstrate that binary black hole populations—in particular, their detection rates, chirp masses, and redshifts—can be used to measure cosmological parameters describing the redshift-dependent star formation rate and metallicity distribution. We present a method that uses artificial neural networks to emulate binary population synthesis computer models, and construct a fast, flexible, parallelizable surrogate model that we use for inference.

Unified Astronomy Thesaurus concepts: [Stellar mass black holes \(1611\)](#); [Gravitational wave astronomy \(675\)](#); [Neural networks \(1933\)](#)

1. Introduction

Rapid binary population synthesis software packages (e.g., BSE (Hurley et al. 2002), StarTrack (Belczynski et al. 2008), `binary_c` (Izzard et al. 2018), SEVN (Iorio et al. 2022), COMPAS (Team COMPAS et al. 2022a), and cosmic (Breivik et al. 2020)) have a number of free parameters that allow modelers to:

1. set the initial conditions of the simulation (e.g., initial stellar mass, metallicity, separation (for binary stars), etc.), and
2. determine the nature of some of the (simulated) physical processes that a star, single or binary, undergoes during its evolution through time (e.g., wind mass-loss rate, supernova remnant masses and kicks, and mass transfer and common envelope efficiency for binary stars).

Many of these free parameters and the physical states and processes they represent are currently not well constrained by observation.

Our ultimate goal is to develop a method to determine constraints for (some of) the physical states and processes modeled by various modeling and simulation software packages, thus not only learning more about the physical nature of these states and processes, but also improving our ability to model them. A naïve, brute-force method to determine such constraints is to generate a synthetic state space by integrating over all possible values of initial conditions and evolutionary parameters, then search that synthetic state space for states that correspond to observed data. However, even an indicative selection of that parameter space is very large (e.g., Broekgaarden et al. 2022 considered 560 distinct model variations), and generating such a large state space with current modeling tools is computationally intractable.

As a first step toward achieving our goal, we develop a tool that can quickly simulate a state space large enough to allow us

to infer astrophysical constraints. We can do that by generating a smaller number of states using a targeted selection of initial conditions and evolutionary parameters, and then using that simulated state space as a set of training examples to teach an interpolant how to map initial conditions and evolutionary parameters to the final state, thus avoiding the computational time of calculating the intermediate steps. Once we have a working tool, we can then generate a very large simulated state space and search that state space for states that match observed data. Here, as proof of the concept, we describe the construction of the tool for a reduced space of parameters: four parameters governing the cosmic metallicity-specific star formation rate (MSSFR) in the model of Neijssel et al. (2019).

The remainder of this paper is organized as follows. Section 2 presents a description of the tool we construct for this proof-of-concept study, as well as the method used to train the tool. We present and discuss our results in Section 3. We provide some concluding remarks in Section 4.

2. Method

2.1. Overview

The COMPAS population synthesis software suite (Stevenson et al. 2017; Vigna-Gómez et al. 2018; Team COMPAS et al. 2022a) allows users to generate synthetic populations of binary systems. The COMPAS suite includes post-processing tools that allow users to, among other things, calculate merger rates, detection rates of mergers, etc., for double compact objects (DCOs) that will merge within the age of the Universe (Barrett et al. 2018; Neijssel et al. 2019; Team COMPAS et al. 2022b).

For this proof-of-concept study, we develop a surrogate model for COMPAS that provides fast predictions of detection rates of merging DCOs as a function of chirp mass and merger redshift. Surrogate models have been used to emulate various aspects of the simulated evolution of binary systems and to infer constraints on simulation parameters, including the evolution of individual binary systems (e.g., Lin et al. 2021, using a Gaussian process emulator (Rasmussen & Williams 2005) and classifier) and populations of binaries (e.g., Barrett et al. 2016; Taylor & Gerosa 2018, Wong & Gerosa 2019; Cheung et al. 2022, using Gaussian process emulators or

normalizing flows, and Wong et al. 2021 using deep-learning-enhanced hierarchical Bayesian analysis).

We chose detection rates because these are calculated by the COMPAS post-processing tools after the evolution of the population of binary systems by convolving binary evolution outcomes with the MSSFR. This makes generating example (training) data for our surrogate model by varying MSSFR parameters much less onerous than it would be if we needed to evolve new populations of binaries whenever our astrophysical parameters are varied.

We construct an interpolant that, given a set of input values, predicts a detection rate matrix for the merger of DCOs. Input to the interpolant is a set of values for four MSSFR parameters chosen for this study (see Section 2.2). The interpolant outputs detection rates binned into a matrix of 113 chirp mass bins by 15 redshift bins (see Section 2.3). The detection rates generated by the interpolant are combined rate predictions for all DCOs: binary black holes, binary neutron stars, and mixed neutron star—black hole binaries. The interpolant we construct is comprised of a matrix of simple feed-forward artificial neural networks (ANNs), with each ANN trained to predict the detection rate for a single cell of the resultant detection rate matrix.

We first simulate the evolution of 512 million binary systems using COMPAS, resulting in a population of approximately 1.3 million DCOs that merge within the age of the Universe, ~ 13.8 Gyr.

The COMPAS modeling tool has many configurable options that allow the user to change initial conditions and evolutionary parameters. The simulation for this study was performed using the COMPAS fiducial values for all configurable options (see Appendix B; described in detail in Team COMPAS et al. 2022a), and for our purposes we consider the population of binary systems simulated by COMPAS to be indicative of the population of binary systems in the Universe.

The COMPAS suite includes tools to take a population of binary systems produced by COMPAS, integrate over the known initial condition distribution and the cosmological star formation history, and produce a population of merging DCOs with individual masses and merger redshifts, to which observational selection effects can be applied, converting the population into a prediction for the observable merger rate over the population of DCOs.

We construct a grid of six different values for each of the four MSSFR parameters under study (see Section 2.2), and for each entry in the grid we use the COMPAS tools described above to calculate a detection rate matrix for the population of merging DCOs, resulting in 1296 detection rate matrices, which we use to create the training data for the ANNs that comprise our interpolant. The calculated detection rate matrices contain, for each bin in the chirp mass–redshift space, the expected detection rate (detections per year) for an interferometer network with an indicative sensitivity of the LIGO O3 run (Abbott et al. 2020).

For each cell in the detection rate matrix, we construct and train an ANN that predicts the detection rate for that cell, given values for the four MSSFR parameters—this matrix of ANNs comprises our interpolant.³ We chose a matrix of simple ANNs rather than a single, large ANN for several reasons:

1. Training a large ANN to learn >1500 relationships is difficult and time-consuming, whereas training a small ANN to learn a single relationship is easier and faster.
2. A large ANN is likely to be less accurate, because the network will try to get most relationships close, whereas the single, small ANN just needs to get one relationship right.
3. Overfitting in a large ANN learning multiple relationships is more likely than a single, small ANN learning a single relationship. It is likely that some of the >1500 different relationships to be learned will be easier to learn than others. Training a large ANN to learn multiple relationships could result in the relationships that are easier to learn being overfitted because the ANN needs to be trained for longer to learn the more difficult relationships.
4. ANNs for different cells can have different architectures—some relationships might require more, or fewer, nodes and/or layers.
5. While other global surrogate models could provide computational cost savings by utilizing the regularity (smoothness) of the output over the input parameter space, assumptions of regularity are risky when some astrophysical parameters could lead to bifurcations in the outputs.
6. We can replace the ANN for some cells with a different model without affecting all cells—some relationships may not need to be modeled by an ANN: a simple function fit (possibly even a constant value) might be sufficient.
7. We can retrain the ANNs for individual cells as necessary, without the need to retrain all cells.
8. We can improve performance by running individual ANNs in parallel on different CPUs.

2.2. MSSFR Parameters

The parameters we chose to vary for this study are four free parameters for the calculation of the MSSFR in the phenomenological model of Neijssel et al. (2019). In this model, the MSSFR is split into two parts, namely the star formation rate (SFR) and the metallicity distribution:

$$\frac{d^3 M_{\text{SFR}}}{dt_s dV_c dZ}(z) = \frac{d^2 M_{\text{SFR}}}{dt_s dV_c}(z) \times \frac{dP}{dZ}(z), \quad (1)$$

where the SFR is given by

$$\frac{d^2 M_{\text{SFR}}}{dt_s dV_c} = a \frac{(1+z)^b}{1 + \left(\frac{1+z}{c}\right)^d} M_{\odot} \text{yr}^{-1} \text{Mpc}^{-3}. \quad (2)$$

We use a log-normal distribution in metallicity at each redshift (cf. the skewed log-normal model proposed by van Son et al. 2022),

$$\frac{dP}{dZ}(z) = \frac{1}{Z\sigma\sqrt{2\pi}} e^{-\frac{(\ln(Z) - \ln(\langle Z \rangle) + \sigma^2/2)^2}{2\sigma^2}}, \quad (3)$$

with a redshift-independent standard deviation σ in $\ln(Z)$ space around a redshift-dependent mean μ of $\ln(Z)$ given by

$$\langle Z \rangle = e^{\left(\mu + \frac{\sigma^2}{2}\right)}, \quad (4)$$

³ Some cells in the detection matrix may not require the use of an ANN—see Section 2.4.

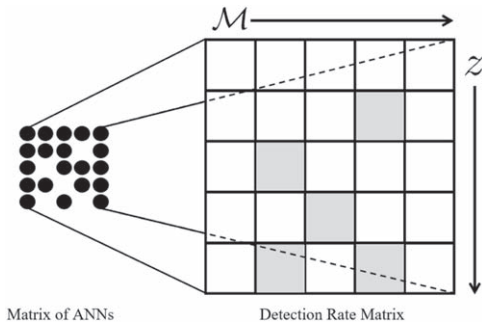


Figure 1. The interpolant.

with mean metallicity parameterized as in Langer & Norman (2006):

$$\langle Z(z) \rangle = Z_0 10^{\alpha z} \quad (5)$$

We vary SFR parameters a and d , as well as the metallicity distribution parameters α and σ , in this study, while fixing $b = 2.77$, $c = 2.9$, and $Z_0 = 0.035$. Neijssel et al. (2019) demonstrated that a range of star formation history models can be mimicked by varying only this subset of MSSFR parameters. We therefore choose to vary the same parameters for consistency with their analysis. Moreover, reducing the number of parameters allows us to limit computational complexity for this proof-of-principle study.

2.3. Binned Rates

As noted in Section 2.1, the detection rates generated by our interpolant are binned into a matrix of 113 chirp mass bins by 15 redshift bins.

The redshift bins are constructed as fixed-width bins, with the lower edge of the first bin at redshift 0.0 and each bin having a width of 0.1—the upper edge of the final (15th) bin is at redshift 1.5.

The chirp mass bins are constructed as variable-width bins, with the lower edge of the first bin at $L_1 = 0.0 M_\odot$ and the upper edge at $L_2 = 0.5 M_\odot$. The remaining chirp mass bin edges fall at

$$L_{i+1} = \frac{41}{39} L_i, \quad 2 \leq i \leq 112. \quad (6)$$

Thus, the width of the 2nd to 112th bins is equal to 5% of the median chirp mass for the bin. The final (113th) bin extends to infinity.

2.4. The Interpolant

As noted in Section 2.1, the interpolant we construct is comprised of a matrix of ANNs. For this study, all ANNs in the matrix have identical architecture, though that is not a requirement of the method: each ANN is independent of the others, and the architecture of individual ANNs could be varied. Nominally, there will be a corresponding ANN for each cell in the detection rate matrix, but there may be some cells for which an ANN is not required. For example, if a given cell in the detection rate matrix is known to be a constant value (e.g., zero), training an ANN is neither useful nor necessary—we can instead replace the ANN for that cell with, e.g., a simple equation and skip training an ANN. A schematic diagram of the interpolant is shown in Figure 1. In

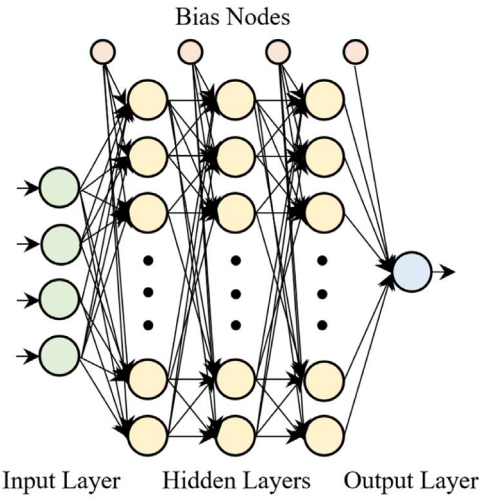


Figure 2. ANN architecture.

Figure 1, ANNs are trained to predict values only for the unshaded cells in the detection rate matrix; values for the shaded cells will be predicted using some other model, as discussed above.

2.4.1. The Artificial Neural Networks

Each ANN is a fully connected feed-forward network of artificial neurons (nodes), with an input layer, three hidden layers, and an output layer. The input layer consists of four input nodes (one for each MSSFR parameter), the output layer consists of a single node (to indicate the predicted detection rate for the corresponding matrix element), and each of the three hidden layers contains 64 nodes. The hidden and output layers have an associated bias node that provides each node in the respective layer with a bias weight. A schematic diagram of a representative ANN is shown in Figure 2.

While there are some generally accepted rules of thumb, the selection of ANN architecture is very often arbitrary. In this case, the architecture was chosen to produce relatively robust results quickly—no attempt was made to optimize the architecture for the best performance, with respect to either accuracy or speed. The input layer and output layer are fixed (the four MSSFR parameters are the inputs, and since the ANN is a regressor, not a classifier, we have a single, continuous, output node), and the choice of three hidden layers, each with 64 nodes, was a brute-force rather than finessed approach. Per the universal approximation theorem,⁴ a shallow neural network (with a single, wide, hidden layer) can approximate any function—after a little experimentation with different architectures, the architecture described above was found to produce good enough results reasonably quickly. While there is no universally accepted rule as to how many layers constitute a deep ANN (how many grains make a heap?⁵), the architecture implemented here probably qualifies as a deep network (because the networks have more than two hidden layers), but it is essentially a simple feed-forward ANN (albeit, given the number of nodes on each layer, a large simple feed-forward ANN).

⁴ https://en.wikipedia.org/wiki/universal_approximation_theorem

⁵ https://en.wikipedia.org/wiki/Sorites_paradox

Table 1
 λ Training Values

λ_x	Values
α	[−0.500, −0.400, −0.300, −0.200, −0.100, −0.001]
σ	[0.100, 0.200, 0.300, 0.400, 0.500, 0.600]
a_{SF}	[0.005, 0.007, 0.009, 0.011, 0.013, 0.015]
d_{SF}	[4.200, 4.400, 4.600, 4.800, 5.000, 5.200]

2.4.2. Training

Training an ANN requires many examples of the relationship (s) to be learned (e.g., example outputs for given inputs)—anywhere from hundreds to millions, depending upon the complexity of the relationship(s) to be learned. By designing the interpolant as a matrix of independent, small ANNs, we have minimized the number and complexity of the relationships to be learned by a single ANN, and as a result, the size of the training data set required is significantly reduced.

The data used to train the ANNs were constructed by using the COMPAS post-processing tools to calculate detection rate matrices for 1296 combinations of the MSSFR parameters under study: six different values for each of the four parameters, collectively labeled λ , as shown in Table 1. The trained interpolant is expected to be able to interpolate between the bounds of each of the constituent parameters of λ .

We chose the ranges of values based on the preferred values from Neijssel et al. (2019), which are the default COMPAS values for each of the parameters—the ranges are a spread around the preferred values.

The question of how many training examples are needed to train an ANN effectively has no simple answer. We need a sample of data that are representative of the problem we are trying to solve (in our case, constructing an interpolant for DCO merger detection rates), and in general, the examples in the sample should be independent and identically distributed. We are training ANNs that we hope will learn to correctly map input data to output data—and moreover, to interpolate and map input data that they have not seen during training. That is, we hope our ANNs will learn the relationships between the input data and output data. We need sufficient training data to reasonably capture the relationships that (we expect) exist both between input features as well as between input features and output features.

The data set described above contains a single example of the 1296 mappings (relationships) that we will present to the ANNs. We know that a single example is almost certainly not sufficient to correctly train the ANNs, and with too few training examples, we risk overfitting the ANNs and possibly compromising their ability to generalize beyond the training data set (but also see Appendix A). In fact, we know that the training data are not perfectly accurate, because the COMPAS binary population represents a Monte Carlo sampler over initial conditions. Therefore, training data suffer from statistical sampling uncertainty. To account for this, we create two new training data sets that we will use to train and test the networks. The results will give us an indication of the number of training examples required. For each new training data set, we use bootstrap sampling of COMPAS outputs to create a number of new detection rate matrices.

The first sampled data set contains 10 different bootstrapped examples for each of the 1296 mappings, resulting in a training data set with 12,960 matrices, and the second contains 100 different bootstrapped examples for each of the 1296 mappings, resulting in a training data set of 129,600 matrices.

As discussed previously, we do not need to train ANNs for cells in the detection rate matrix that we know to contain constant rates. For each training data set, we inspect all training examples and determine which cells in the matrix are constant across all examples—ANNs are not required for those cells. We found that, for the 1695 cells in the detection rate matrix, we only needed to train 292 ANNs for the smaller training data set and 293 ANNs for the larger training data set—for all other cells, the detection rate across all training examples was below our threshold for zero (0.001 detections per year).

Training of the interpolant was conducted using the Keras API (Chollet et al. 2015) for Python (Van Rossum & Drake 2009). Keras is a high-level neural networks API that uses the Tensorflow machine-learning platform (Abadi et al. 2015).

We used a fairly naïve training methodology. Just as no attempt was made to optimize the architecture for the best performance, no real attempt was made to optimize the training for the best accuracy of the networks; “good enough” accuracy was all we were looking for in this proof-of-concept study. We divided our collection of models with different choices of parameters λ and corresponding bootstrapped examples into a training set (80% of all choices of λ) and a validation set (the remaining 20% choices of λ). Networks were trained using the set of training examples, and then tested against the validation set—examples that the network undergoing training had not seen during training. k -fold cross-validation was not performed—some nonexhaustive tests were performed using k -fold cross-validation, with no significant improvement in accuracy, so in the interest of reduced training time, we chose not to implement k -fold cross-validation.

Each network was initialized with random connection weights and biases, then trained for a maximum of 4500 epochs (an epoch is the presentation of the entire training set to the network), after which the network was tested on the validation set. Early stopping was enabled—if no improvement in accuracy (on the training data) was evident after a specified number of epochs (since the last improvement), training was halted.

The network attributes (weights and biases) were adjusted during training using the Keras Adam optimizer (a stochastic gradient descent optimization algorithm), with a custom metric for measuring network performance⁶.

The metric used for measuring the performance of the network was half the Poisson uncertainty of the detection count associated with the bin being predicted, assuming 1 year of observations: where the expected (target) detection count for a given bin is t , a prediction, p , is considered correct if $(t - \frac{\sqrt{t}}{2}) \leq p \leq (t + \frac{\sqrt{t}}{2})$. We used sufficiently large COMPAS binary populations for this study to ensure that the bootstrapping uncertainty is typically small relative to \sqrt{t} : fewer than 0.01% of training and validation samples fall outside a range of width \sqrt{t} . The majority of these outliers were found in specific bins in chirp mass—redshift space with low DCO counts (see Section 3.1.2) and correspondingly high

⁶ <https://keras.io/api/optimizers/adam/>

bootstrapping uncertainty. The accuracy requirement could have been loosened for such bins, but given the low outlier count, we were comfortable with using this definition of performance.

If the network did not achieve 100% accuracy on the validation set, it was trained again with a different set of (random) initial weights and biases. If the network did not achieve 100% accuracy on the validation set after 10 tries at training, the network that achieved the highest accuracy during training was accepted.

2.4.3. Performance

There are three aspects of surrogate model performance that interest us: speed, accuracy, and the ability to generalize (specifically, in our case, to interpolate). If we are to use the interpolant to create large synthetic state spaces that we can explore, we need it to be very fast, at least by comparison with existing methods (e.g., the COMPAS tools described in Section 2.1), to be accurate, and to generalize from the training examples.

The accuracy of the interpolant, with respect to generating a correct detection rate matrix for a given set of MSSFR parameters, is measured during training (see Section 2.4.2). Because the validation data are technically used as part of training—in determining whether training can be stopped or can continue to another iteration—we conducted several ad hoc spot checks of the interpolant against results generated using COMPAS, and in all cases the interpolant matched (within acceptable error bounds) the COMPAS results, thus confirming the ability to generalize/interpolate.

Measuring the speed of the interpolant is straightforward. For this study, because the interpolant consists only of ANNs, the speed of the interpolant is dependent mainly upon the execution (forward propagation) speed of its constituent ANNs and how many ANNs comprise the interpolant—with a small amount of input/output (IO) overhead to read the input data (i.e., the values of λ), present it to the network, and write the output data.

All speed-related performance tests were conducted on a Hewlett Packard ProLiant ML350p Gen8 with two Intel E5-2650v2 processors running at 2.6 GHz, providing 32 threads (with hyper-threading) and 96 GB of RAM—performance figures quoted are for that system.

The total elapsed time to create both training data sets was ~ 142 hr.

In this study, the ANNs that comprise our interpolant have identical architectures. Therefore, if we ignore the IO overhead, the speed of the interpolant is directly proportional to the execution speed of a single ANN. Training of the ANNs that comprise the interpolant is independent, and therefore can be done in parallel. We measured the average execution speed of a single ANN over 1,000,000 executions to be 2.19×10^{-5} s per execution. If we consider the interpolant to be the serial execution of its (for this study) 293 constituent ANNs (292 for the smaller data set), the execution speed of the interpolant (ignoring IO overhead) is $2.19 \times 10^{-5} \times 293 = 6.42 \times 10^{-3}$ s. However, because the execution of the constituent ANNs is independent, the architecture of the interpolant lends itself to parallel execution (it is, in fact, embarrassingly parallel), so the speed of execution of the interpolant could be reduced significantly by spreading the execution of the constituent ANNs across multiple CPUs/cores.

The total elapsed time to train all ANNs was ~ 1.5 hr for the smaller training data set and ~ 12 hr for the larger training data set.

As noted earlier, our interpolant is only useful if it is significantly faster than existing methods. We measured the average execution speed of the COMPAS toolset to create a merger detection rate matrix from a population of DCOs over 50 executions to be 111.97 s. The COMPAS post-processing toolset (to create the detection rate matrix) is not parallelizable.

Our interpolant, if run in serial fashion, is $\sim 17,500$ times faster than the COMPAS toolset, and naively, it could be up to ~ 5 million times faster if run in a fully parallel fashion (though there would be some overhead in running in parallel). We should note that the COMPAS post-processing tools are written in Python (but it is unlikely that any optimization would result in improvements of more than tens of percent).

2.5. Searching the State Space

With the interpolant constructed, we shift our focus to showing that the interpolant enables us to infer constraints on our chosen MSSFR parameters. To do this, we search the state space over the range of values of the parameters we used to train our interpolant (the interpolant is not guaranteed to be valid outside those ranges) and seek out models that match the observed data.

First, to validate the method and test the interpolant, we create a mock data set that, for this test, we consider to be representative of the Universe. We use COMPAS as our forward model to create a population of DCOs representative of the Universe, and as noted earlier, we use the COMPAS fiducial values for all configurable options.

We use the COMPAS post-processing tools to create mock “true” data, \mathcal{D} , as follows. We specify a detector noise spectrum (equivalent to an approximate LIGO O3 sensitivity) and choose “true” values of the MSSFR parameters λ . We use the COMPAS tools to generate a DCO merger detection rate matrix that represents the Universe defined by the COMPAS fiducial model. We multiply the detection rate matrix by a fixed run duration to produce a matrix of expected detections, and then independently sample mock observations from each chirp mass—redshift in this matrix following Poisson statistics.

We then use our surrogate model to generate a detection rate matrix at various points in the state space (characterized by varying λ), multiply this by the same run duration, and calculate the likelihood of observing \mathcal{D} given λ at those points as described below. This produces a “likelihood landscape” that we can search. Finding the maximum likelihood on the landscape tells us the value of λ (the interpolant considers is) most likely to produce our “true” data. The likelihood surface (or in Bayesian language, the posterior on the parameters λ under the assumption of flat priors over the range considered) contains information about uncertainty in the MSSFR parameter inference and any correlations between these parameters.

Results of the validation of the method are presented in Section 3.2.1.

2.5.1. Likelihood Calculation

We define the likelihood \mathcal{L} that we will see the data \mathcal{D} for a particular set of parameters λ as

$$\log \mathcal{L}(\mathcal{D}|\lambda) = \log \mathcal{L}(N_{\text{obs}}|\lambda) + \sum_{i=1}^{N_{\text{obs}}} \log p(D_i|\lambda), \quad (7)$$

where N_{obs} is the number of observations in the data $\mathcal{D} \equiv \{D_1, D_2, \dots, D_{N_{\text{obs}}}\}$.

The first term of Equation (7) is a Poisson likelihood on there being N_{obs} detections,

$$\log \mathcal{L}(N_{\text{obs}}|\lambda) = N_{\text{obs}} \log(\mu(\lambda)) - \mu(\lambda), \quad (8)$$

where $\mu(\lambda)$ is the expected number of observations and we omit terms that depend on the data only and therefore disappear on normalization, such as $\log(N_{\text{obs}}!)$ and permutation coefficients.

The second term of Equation (7) is comprised of a product of the probabilities of individual detections,

$$p(D_i|\lambda) = p(z = z_i, \mathcal{M} = \mathcal{M}_i|\lambda), \quad (9)$$

where $p(z = z_i, \mathcal{M} = \mathcal{M}_i|\lambda)$ is the entry in the probability distribution matrix $p(z, \mathcal{M}|\lambda)$ in the z and \mathcal{M} bin of the observed i^{th} event. The probability distribution matrix $p(z, \mathcal{M}|\lambda)$ is constructed by dividing the entries in the matrix of expected detections for a given λ , generated by multiplying the surrogate model detection rate matrix by the run duration, by the sum of the entries in the matrix, μ .

2.6. Using Gravitational Wave Observations

The catalog of LIGO, Virgo, and KAGRA (LVK) gravitational wave observations (LIGO Scientific & Virgo Collaborations 2022, LIGO Scientific & Virgo & KAGRA Collaborations 2021) provides us with a number of merger detections from a population of DCOs. For each LVK event, we know, among other things, the chirp mass of the merging binary and the merger redshift, up to a measurement uncertainty. We only include confident detections with a minimum astrophysical probability $p_{\text{astro}} \geq 0.95$; although less confident events could be included (e.g., Farr et al. 2015), they will typically contribute little information, due to greater measurement uncertainties, so we omit them for this proof-of-concept study.

Once we validate our method, we can use our surrogate model and the LVK data to infer real physical constraints on the parameters under study, with the important caveat that we have fixed the astrophysical parameters, and any errors in those could bias the inferred MSSFR model values.

2.6.1. LVK Likelihood Calculation

In practice, for LVK events, we may not know the values z_i , \mathcal{M}_i and hence the probability $p(z = z_i, \mathcal{M} = \mathcal{M}_i|\lambda)$ for the observed i^{th} event in Equation (9) perfectly, but instead may only have K samples from a posterior $p(z_i, M_{c,i}|D_i)$. For LVK data, we substitute the following for Equation (9):

$$p(D_i|\lambda) = \frac{1}{K} \sum_{k=1}^K \frac{p(z = z_i^k, M_c = M_{c,i}^k|\lambda)}{\pi(z_i^k, M_{c,i}^k)}, \quad (10)$$

where the subscript k refers to the k^{th} posterior sample among K available samples for event i and $\pi(z, M_c)$ is the prior used by LVK data analysts in the original inference on the observed events (see, e.g., Mandel et al. 2019).

3. Results and Discussion

3.1. Interpolant Training Results

The detection rate matrix generated by our interpolant consists of 1695 cells: 113 chirp mass bins x 15 redshift bins. Of the 1695 cells in the matrix, only 17% (292 for the smaller

Table 2
ANN Training Statistics

Training Statistic	#Examples	
	10	100
Lowest train time (min)	0.26	1.37
Highest train time (min)	42.98	301.37
Average train time (min)	9.85	74.1
Average tries to train	5.76	7.18
Fewest tries to train	1	1
Most tries to train	10	10
Number of networks trained	292	293

training data set and 293 for the larger training data set) require an ANN to predict the rate for that cell: all other cells have zero rates for all training examples. The time to train the interpolant and the accuracy achieved during training are shown below.

3.1.1. Training Times

Details of the training times for the ANNs trained are presented in Table 2. As noted in Section 2.4.3, the total elapsed time to train all ANNs was ~ 1.5 hr for the smaller training data set and ~ 12 hr for the larger training data set.

3.1.2. Interpolant Accuracy

As discussed in Section 2.4.2, each of the constituent ANNs of the interpolant was trained until an accuracy threshold was reached—with accuracy being measured as the percentage of the validation data set the ANN was able to correctly predict (the meaning of “correct” is explained in Section 2.4.2).

Details of the interpolant accuracy achieved during training are presented in Table 3.

The accuracy figures in Table 3 show that, for both training data sets, the average accuracy achieved by the ANNs trained was better than 99.4%, with more than 90% of the ANNs achieving an accuracy of 99% or better.

Figure 3 shows in detail the accuracy achieved for each of the ANNs—the grid shown corresponds to the detection matrix, with each cell representing the ANN corresponding to a $\{z, \mathcal{M}\}$ bin.

From the top two panels of Figure 3, we can see that most of the ANNs (for both training data sets) were trained to a high degree of accuracy, with only a few ANNs not achieving 95% accuracy or better. We can inspect the data and the training records for those few ANNs, and because the architecture of the interpolant allows us to easily replace individual constituent ANNs (or whatever model we have used for a particular cell), we can retrain them to achieve a better accuracy, change the ANN architecture, etc. For example, we find that increasing the number of hidden layers from three to five allows us to achieve 100% accuracy across all cells. However, there is a risk of attempting to memorize rather than learn the model with overtrained ANNs (see Appendix A).

The shape and location of the green areas in Figure 3 is interesting. The white cells in Figure 3 are $\{z, \mathcal{M}\}$ bins for which the detection rate was below the threshold for zero (0.001 detections per year). The corollary is that the green areas are $\{z, \mathcal{M}\}$ bins that contain DCOs that are detectable at LIGO

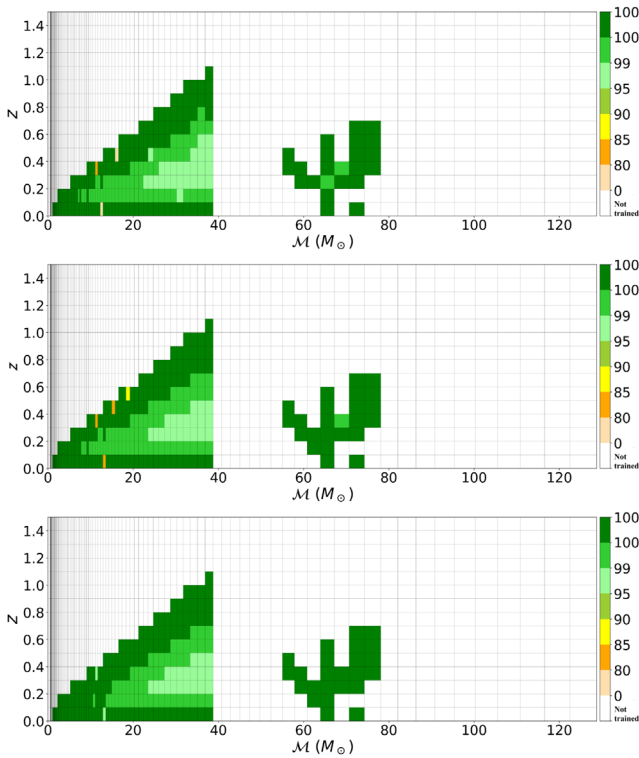


Figure 3. Interpolant training accuracy: 10 training examples (upper panel); 100 training examples (middle panel); 100 training examples with correctness threshold of $|t - p| \leq \max(\sqrt{t}/2, 0.5)$ (bottom panel). White cells represent ANNs that did not require training (zero rates); the meaning of other colors is shown on the color bar.

O3 sensitivity. The triangle on the left is delineated by lower horizon distances for smaller masses along the diagonal line at the top, and by the lower edge of the pair-instability mass gap on the right (Heger & Woosley 2002; Marchant & Moriya 2020).

Many of the bins along the top edge of the triangle (where the detection probability decreases toward zero at the horizon distance) and the bottom edge (only a small comoving volume is contained between redshifts of $z = 0$ and $z = 0.1$) have low detection rates. These bins also contain the apparently poorly performing ANNs in the top two panels. However, this is largely an artifact of our choice of a relative tolerance threshold for correctness, which becomes increasingly stringent as the detection rate decreases. We can instead consider a combination of relative and absolute tolerances, which better accounts for the fact that a detection in a bin is unlikely if the predicted detection rate is low, making excessive precision in the surrogate model unnecessary. If we replace the threshold for correctness of surrogate model prediction p given expected target count detection count t with $|t - p| \leq \max(\sqrt{t}/2, 0.5)$ (cf. the default threshold of $|t - p| \leq \sqrt{t}/2$ from Section 2.4.2), we find that all ANNs achieve an accuracy above 95% (see bottom panel of Figure 3).

We see a green cactus-shaped area to the right of the triangle (that is, to the right of the lower edge of the pair-instability mass gap). To understand this, we need to understand the distribution of DCOs over the chirp mass range—this is shown in Figure 4. Figure 4 shows that almost all of the ≈ 1.3 million merging DCOs in the training data are located in the bins where $\mathcal{M} \lesssim 40M_{\odot}$ (the green triangle in Figure 3), with a few bins in

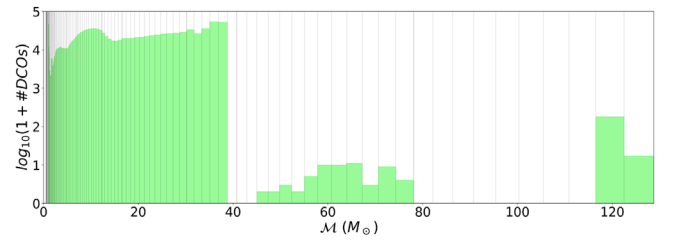


Figure 4. DCO counts for chirp mass \mathcal{M} bins.

Table 3
ANN Training Accuracy

	#Examples	
	10	100
% correct		
Highest	100.0	100.0
Lowest	54.16	83.0
Average	99.44	99.58
Stddev	3.23	1.84
Counts		
% correct = 100	139	97
$99 \leq \% \text{ correct} < 100$	124	172
$95 \leq \% \text{ correct} < 99$	26	20
$90 \leq \% \text{ correct} < 95$	0	0
$85 \leq \% \text{ correct} < 90$	0	1
$80 \leq \% \text{ correct} < 85$	1	3
% correct < 80	2	0
Number of networks evaluated	292	293

the $\approx 50\text{--}80M_{\odot}$ range (the green cactus-shaped area) with 10 or fewer DCOs per bin, and then two bins at $\approx 120M_{\odot}$ with a total of ≈ 190 DCOs (not seen in Figure 3, as the detection rates for these bins were below the zero threshold).

Further inspection of the data reveals that, of the ≈ 1.3 million merging DCOs, only 50 are located in the $\approx 50\text{--}80M_{\odot}$ chirp mass range, while a further 193 are in the two chirp mass bins at $\approx 120M_{\odot}$, and all of these binaries were BBHs comprised of two massive stars that began their evolution on the main sequence as chemically homogeneous stars (Riley et al. 2021). Of the 50 binaries in the $\approx 50\text{--}80M_{\odot}$ chirp mass range, 23 had both stars evolve as chemically homogeneous stars for the entirety of their lifetime on the main sequence, while for the remaining 27, the secondary star did not evolve as a chemically homogeneous star for its main-sequence lifetime. Of the 193 binaries in the $\approx 120M_{\odot}$ chirp mass range, all but one had both stars evolve as chemically homogeneous stars for the entirety of their lifetime on the main sequence—the secondary star of the remaining binary did not evolve as a chemically homogeneous star for its main-sequence lifetime.

With only 243 of the ≈ 1.3 million DCOs produced from a population of 512 million binaries above the lower edge of the pair-instability mass gap, it is clear that these DCOs are very rare (Marchant et al. 2016; Riley et al. 2021). The cactus shape

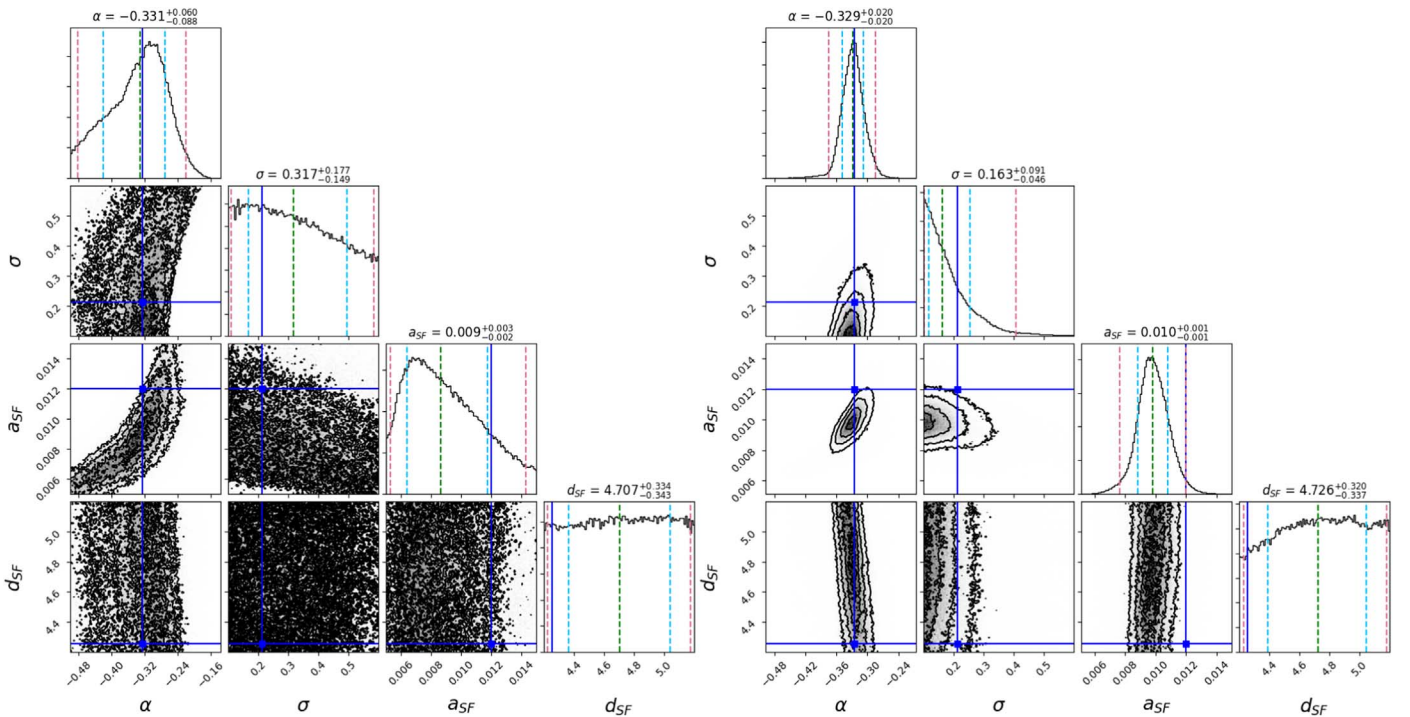


Figure 5. Corner plots showing MSSFR parameter inference on mock observations without measurement uncertainties. The marginalized 1D histograms for each MSSFR parameter appear in the panels along the diagonal, and the marginalized 2D projections of the posterior probability distributions in the off-diagonal panels. The left panel is for the smaller data set (58 events); the right panel is for the larger data set (578 events). The blue solid line indicates the best-fit value, the green dashed line shows the median, and the pale blue and pale red dashed lines show the 68% and 95% credible intervals, respectively. Labels on the diagonals show the mean and 68% credible interval.

in Figure 3 is thus determined by the delay times of chemically homogeneously evolving DCOs and convolution with the MSSFR, with the detailed features likely an artifact of limited sampling.

3.2. Inference with Surrogate Model

3.2.1. Method Validation: Inference on Perfect Measurements

In Section 2.5, we outlined how we validate our method. Here, we present the results of that validation.

Using COMPAS post-processing tools and the synthesized population of 512 million binaries, we created a detection rate matrix, binned as described in Section 2.3, for a known value of λ : $\lambda(\alpha, \sigma, a_{SF}, d_{SF}) = (-0.325, 0.213, 0.012, 4.253)$. Using that detection rate matrix and assuming an observing time of either 0.1 yr or 1 yr, we created mock data sets \mathcal{D} with perfect parameter measurement accuracy for each of the “detected” sources (58 in the 0.1 yr data set, 578 in the 1 yr data set).

We carried out a Markov Chain Monte Carlo (MCMC) search (e.g., Andrieu et al. 2003), using the *emcee* Python package (Foreman-Mackey et al. 2013), over the $\lambda = \{\alpha, \sigma, a_{SF}, d_{SF}\}$ parameter space, assuming flat priors on λ . This search calculates the likelihood $\mathcal{L}(\mathcal{D}|\lambda_i)$ at each λ_i visited by the MCMC algorithm using the surrogate model (see Section 2.5.1 for the likelihood calculation). To confirm that our inference step was performing adequately, we also performed a fine-grained grid search consisting of 51 equidistant values for each of α , σ , and a_{SF} , and 101 equidistant values for d_{SF} , for a total of $51 \times 51 \times 51 \times 101 \approx 13.4$ million points evaluated, followed by a naïve hill-climbing search (e.g., Russell & Norvig 1995), using the results of the grid search as a starting point. We confirmed that

the searches found the same maximum likelihoods (within expected sampling variations).

The MCMC search posteriors are shown in Figure 5. The true value is found within 68% credible intervals for most MSSFR parameters under study, and within 95% credible intervals for all parameters. As expected, the larger data set produces more precise inference on MSSFR parameters, with the ratio of posterior width for α approaching a factor of $\sqrt{10}$ narrower on the data set with 10 times more data (right panel), though the improvement is much smaller for poorly measured parameters, such as d_{SF} , whose posteriors are dominated by the priors.

We considered surrogate models trained on data sets with 10 bootstrapped samples per λ value and 100 bootstrapped samples and found that they are not significantly different, which is consistent with the accuracy of the surrogate model shown in Figure 3. We show results for surrogate models trained with 100 bootstrapped samples per λ from here on.

3.2.2. Method Validation: Inference on Uncertain Measurements

We used the same detection rate matrix as in the previous subsection, corresponding to $\lambda(\alpha, \sigma, a_{SF}, d_{SF}) = (-0.325, 0.213, 0.012, 4.253)$, to validate inference on mock observations with measurement uncertainties. Using that detection rate matrix, and assuming an observing time of one year, we created a data set of mock LVK data containing 578 events. We then randomly sampled 58 events from that data set, creating a new data set with an observing time of 0.1 yr.

We then replaced each of the chosen events with samples from an associated mock posterior. To create these samples, we used a mock model of the LVK prior. We built the source-frame chirp mass prior by assuming that the component masses

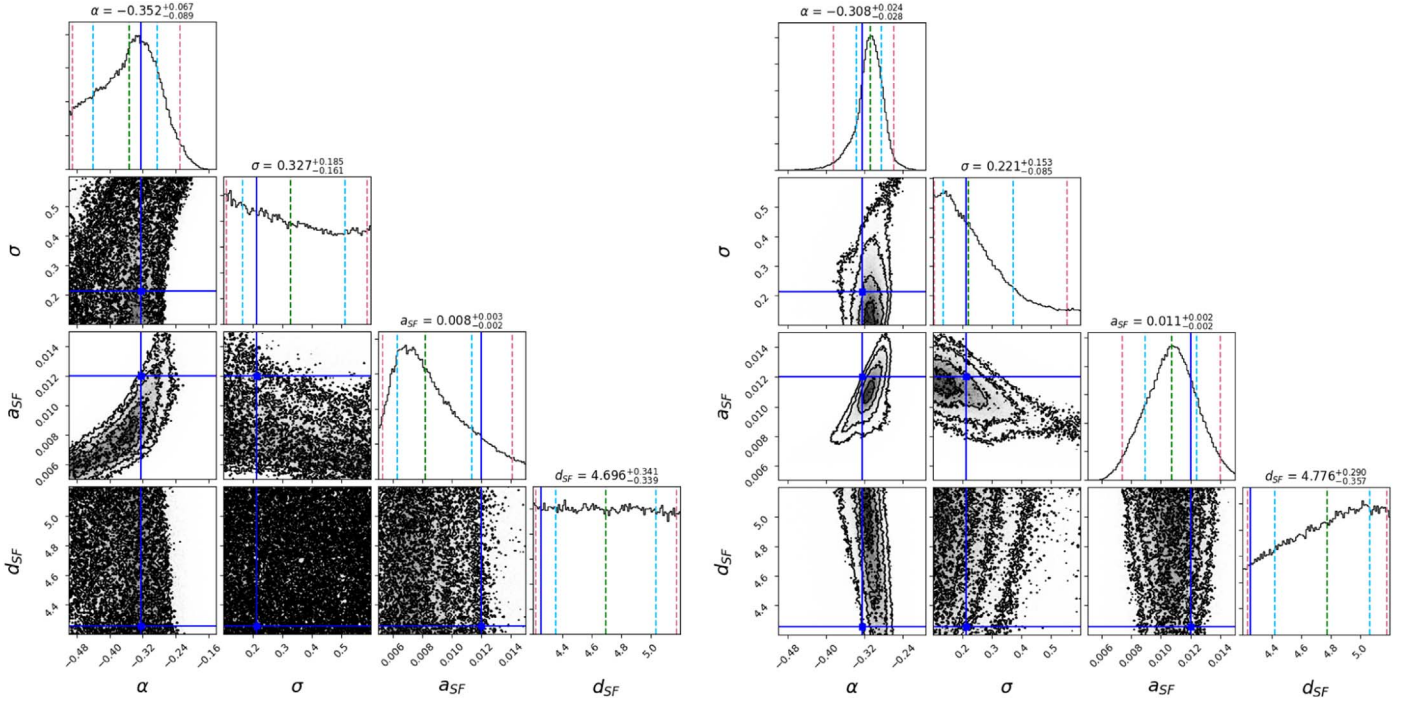


Figure 6. Corner plots for MSSFR parameter inference on mock observations with measurement uncertainty. The left panel is for the smaller data set (58 events); the right panel is for the larger data set (578 events). The blue solid line indicates the “true” value, the green dashed line shows the median, and the pale blue and pale red dashed lines show the 1σ and 2σ credible intervals, respectively.

$m_1 > m_2$ are uniformly drawn from the range $[11,000]M_\odot$, with additional cuts $m_2 \in [0.05m_1, m_1]$ and $\mathcal{M} \equiv m_1^{0.6} m_2^{0.6} (m_1 + m_2)^{-0.2} \in [1200]M_\odot$. For the mock redshift prior, we used $\pi(z) \propto z^2$ on $z \in [0.01, 1.5]$. We then weighed mock chirp mass and redshift samples taken from

$$\begin{aligned} \mathcal{M} &\sim \mathcal{M}^T \left(1 + 0.03 \frac{12}{\rho} (r_0 + r) \right); \\ z &\sim z^T \left(1 + 0.3 \frac{12}{\rho} (r_0 + r) \right) \end{aligned} \quad (11)$$

by these priors. In Equation (11), which follows Powell et al. (2019), the superscript T denotes the true value, ρ is the signal-to-noise ratio sampled from $p(\rho) \propto \rho^{-4}$ with a minimum of $\rho \geq 12$, r_0 is a normal random variable that stochastically shifts the peak of the posterior away from the truth, and r is a vector of normal random variables that provides the spread of the posterior. Thus, the small and large data sets are composed of 58 and 578 events, respectively, each represented by a set of mock posterior samples that contain mock measurement uncertainties on the observed parameters \mathcal{M} and z .

Using each of those data sets as “true” data, we used our surrogate model to infer posteriors on λ , assuming flat priors. The MCMC search statistics are shown in Figure 6.

Figure 6 shows that, as expected, inference provides consistent credible intervals, with most “true” values falling within the 1σ credible interval, and the remainder within the 2σ credible interval. As before, the larger data set produces more precise inference on MSSFR parameters. Comparing Figure 6 with Figure 5, we see that mock measurement uncertainties have limited impact on MSSFR inference for the smaller data set, where inference is predominantly limited by the total number of events. For the larger data set, incorporating

measurement uncertainty does lead to a moderate deterioration in the accuracy of inference.

3.2.3. Inference from Real LVK Observations

Since the goal of this work is to (eventually) develop a method to allow us to infer constraints for (some of) the astrophysical states and processes, as an illustration of the ability of this methodology to assist in inferring constraints on cosmological and astrophysical parameters from real data, we performed an MCMC search (as described in Section 3.2.1) on actual LVK observations from observing runs 3a and 3b (run duration 275.3 days), where $p_{\text{astro}} \geq 0.95$ (52 events).

Here, we present the results of this analysis. However, a few disclaimers are warranted. We only varied parameters in an MSSFR model, while fixing the assumptions and parameters describing stellar and binary evolution (mass transfer, winds, supernovae, etc.) to fiducial—but likely incorrect—values. We also used a simple approximation for the selection function imposed by LVK observations, picking a sample noise spectrum rather than a variable one and using a single-detector signal-to-noise ratio threshold of eight, independent of the source parameters, as a proxy for detectability by the network. A more realistic selection function would be based on an actual injection campaign into the detector noise as accumulated over the third LVK observing run.

The posteriors on the MSSFR parameters on the LVK O3 data are shown in the left panel of Figure 7. The highest-likelihood point found by the MCMC chain is at $\lambda(\alpha, \sigma, a_{\text{SF}}, d_{\text{SF}}) = (-0.085, 0.598, 0.005, 5.042)$. It is clear that the posteriors rail against the prior boundary on a_{SF} .

We therefore repeated the analysis by extending the uniform prior on a_{SF} to incorporate the range $[0, 0.015]$. Fortunately, a_{SF} is a scaling factor only, so this did not require retraining the neural network. We used the surrogate model prediction at the

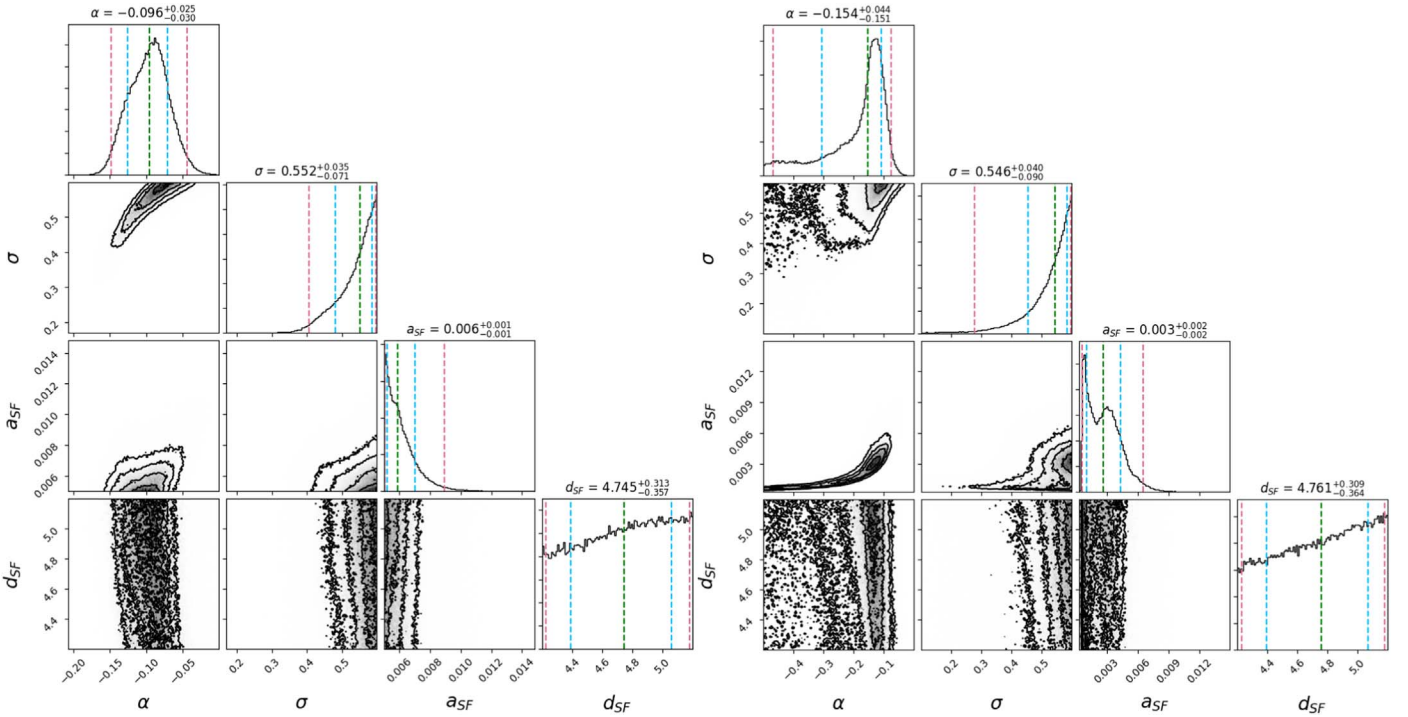


Figure 7. Corner plot for MSSFR parameter inference on LVK data with $p_{\text{astro}} \geq 0.95$ using the original priors (left panel) and using the extended prior on a_{SF} (right panel).

midpoint of the a_{SF} range over which it was trained to predict the detection rate matrix at values of a_{SF} outside this range. Specifically, we assumed that the surrogate model for the detection rate matrix at $a_{\text{SF}} < 0.005$ was equal to $(a_{\text{SF}}/0.01)$ times the prediction of the surrogate model at $\{a_{\text{SF}} = 0.01, \lambda'\}$, where $\lambda' \equiv \lambda/a_{\text{SF}}$ refers to all other parameters. We show the posteriors on MSSFR parameters with this extended prior in the right panel of Figure 7. The highest-likelihood point found by the MCMC chain with extended priors is at $(\lambda(\alpha, \sigma, a_{\text{SF}}, d_{\text{SF}})) = (-0.115, 0.599, 0.003, 5.158)$. These posteriors rail against the prior on σ , suggesting that an even broader distribution of metallicities is preferred under this MSSFR model. On the other hand, the preferred value of the normalization of the star formation rate a_{SF} is a factor of ~ 5 lower than the estimate of Madau & Dickinson (2014).

In Figure 8, we show the cumulative density functions (CDFs) for chirp mass (left panel) and redshift (right panel), for both LVK observations and for our best-fit λ (from the analysis with the extended prior on a_{SF}). This provides a visual posterior predictive check. In this case, our preferred MSSFR model fails this check. The paucity of higher-redshift observations in the model prediction likely indicates an overly conservative detectability estimate, probably through the use of a pessimistic threshold on detectable signal-to-noise ratio and/or a pessimistic noise power spectral density estimate. While this problem is relatively easy to fix, the failure to reproduce the observed features of the chirp mass distribution, such as the sharp, narrow peak near $\mathcal{M} \approx 10 M_{\odot}$, speaks to a flawed underlying binary evolution model. As noted above, we did not consider variations in the stellar and binary evolution assumptions and model parameters, as this was only intended as a proof-of-concept exercise.

Moreover, some of the events, in particular the most massive ones, have been conjectured to arise from channels other than isolation binary evolution, which we did not consider here (see,

e.g., Mandel & Farmer 2022 for a review). Other channels may be particularly relevant for higher-mass binaries, including hierarchical dynamical mergers (e.g., Gerosa & Fishbach 2021) or mergers in AGN disks in which black holes could grow by accretion (e.g., Tagawa et al. 2021). We therefore consider an alternative data set in which we remove the five LVK binaries whose median source-frame chirp mass exceed $40M_{\odot}$ and reweigh the posteriors for remaining binaries by imposing a strict chirp mass prior upper limit of $40M_{\odot}$. We also remove any binaries with chirp mass exceeding $40M_{\odot}$ from the forward models. The resulting posteriors on MSSFR parameters (using the extended prior on a_{SF}) and the cumulative posterior predictive check on the chirp mass distribution are shown in Figure 9. The posteriors on MSSFR parameters shift within one standard deviation relative to Figure 7, allowing for a more consistent chirp mass posterior predictive check than in Figure 8, although our model still fails to reproduce the narrow peak near $\mathcal{M} \approx 10 M_{\odot}$. However, this analysis is based on a single model of binary evolution and only considers the impact of MSSFR parameters; complete inference must simultaneously incorporate the parameters governing stellar and binary evolution along with MSSFR models.

3.2.4. Inference Performance

All MCMC searches performed for this study used an ensemble of 10 walkers (single-threaded) and were allowed to run for 100,000 steps per walker, resulting in 1,000,000 steps per search. Each step involves the creation of a new detection rate matrix (for the values of λ visited at that step). We used the ANNs trained on the 100 sample data set, so our surrogate model was comprised of 293 ANNs, with an execution time of $\sim 6.42 \times 10^{-3}$ s (Section 2.4.3).

The elapsed time for each MCMC search using our surrogate model is $\sim 6.42 \times 10^{-3} \times 1,000,000 = \sim 6420$ s (~ 1.8 hr),

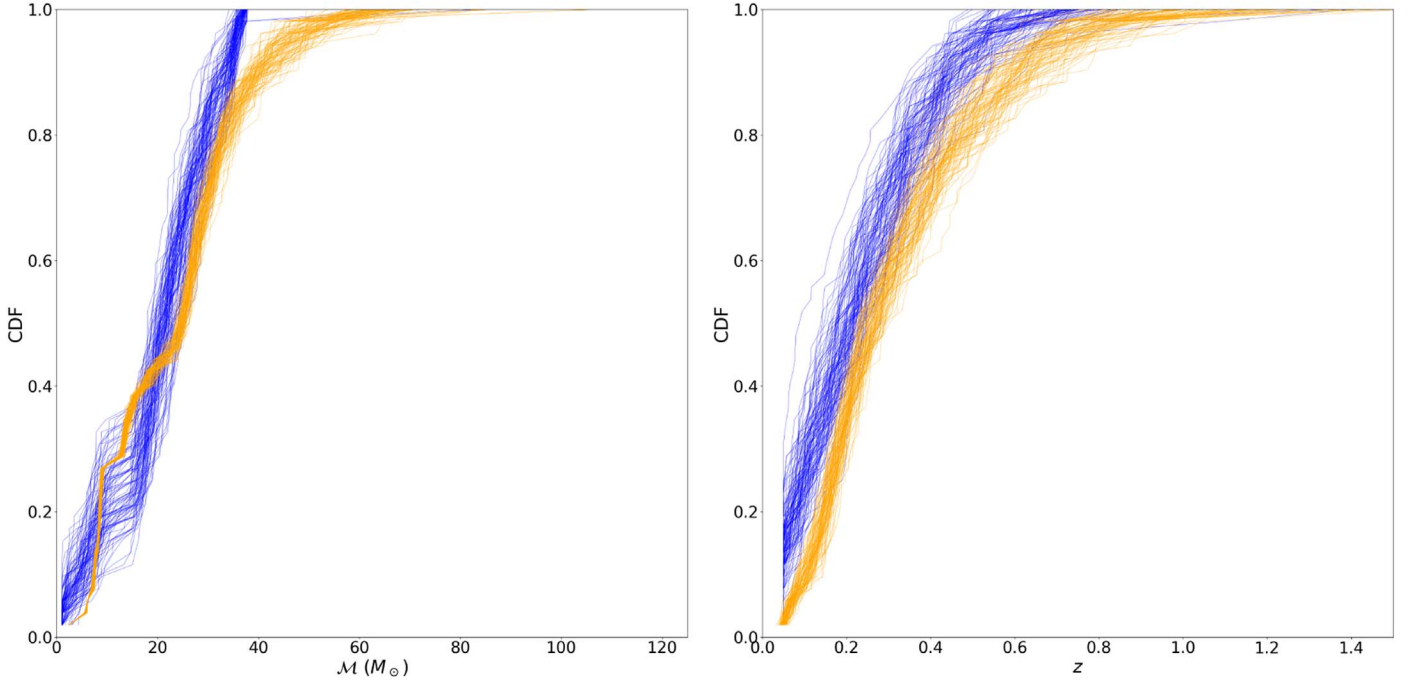


Figure 8. Cumulative density functions for chirp mass (left panel) and redshift (right panel). In each panel, the 100 orange curves are randomly sampled cumulative density functions from the posterior samples (with LVK priors) for each of the DCO mergers detected during the third observing run ($p_{\text{astro}} \geq 0.95$), and they indicate the spread due to measurement uncertainty. The 100 blue curves are cumulative density functions constructed by sampling 52 events from the predicted distribution of detectable DCO mergers from the best-fit λ , and they indicate the sample variance.

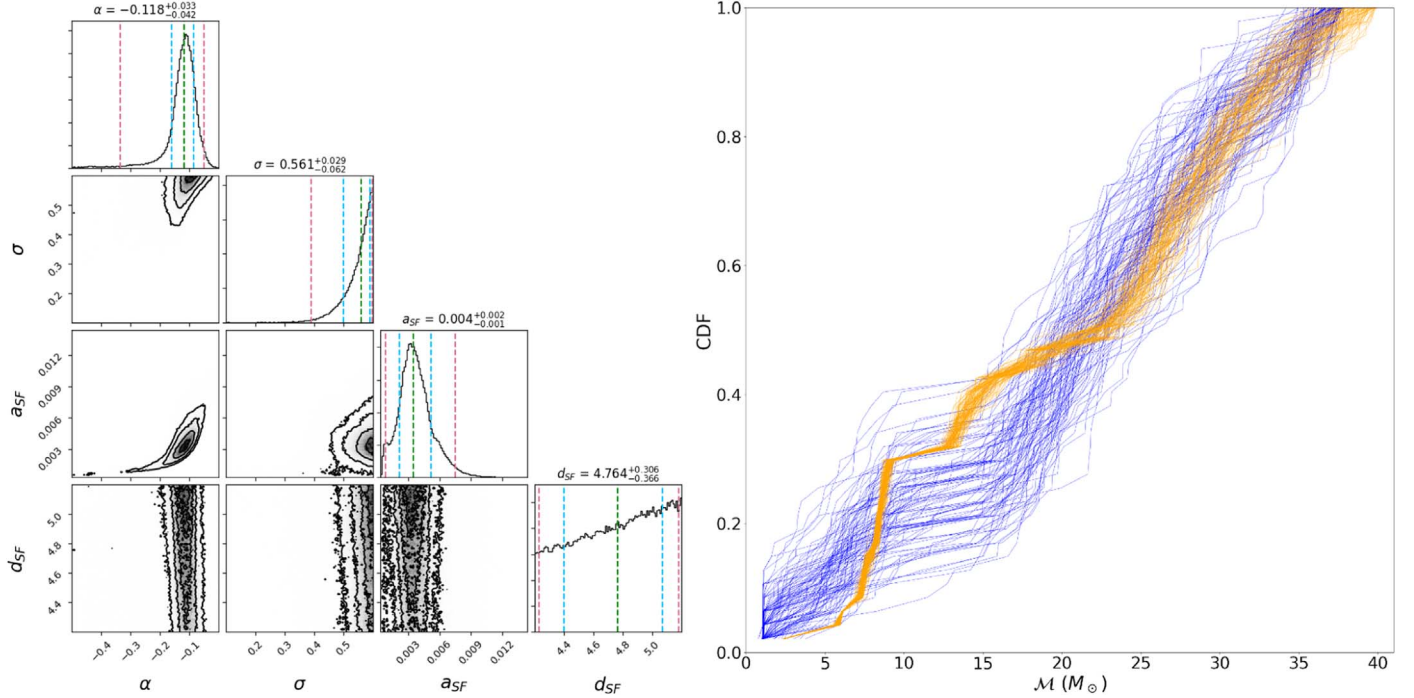


Figure 9. Inference on LVK data after removing the five most massive binaries from the data set (see Section 3.2.3). Left panel: Corner plot for MSSFR parameter inference on LVK data, as in Figure 7. Right panel: posterior predictive check on the chirp mass distribution for the associated best-fit parameters, as in Figure 8.

ignoring MCMC algorithm overheads, which are small compared to the execution time of the ANNs. The time to create all training data sets was 142 hr, and the time to train the ANNs was 12 hr, for a total time 156 hr.

The COMPAS post-processing tools execution time to create a single detection rate matrix is ~ 112 s (Section 2.4.3). The elapsed time for equivalent MCMC searches using the COMPAS

post-processing tools to create the detection rate matrices is $\sim 112 \times 1,000,000 = \sim 112,000,000$ s (~ 3.5 yr), ignoring MCMC algorithm overheads, which are small compared to the execution time of the COMPAS post-processing tools. Thus, the use of surrogate models reduced the total inference cost by a factor of 200 after accounting for the time needed to create training data and train these models.

4. Concluding Remarks

We conducted a proof-of-concept study that showed that it is possible to construct an interpolant that can calculate DCO merger detection rates for a set of astrophysical parameters, with fairly good accuracy, and in milliseconds (several orders of magnitude faster than methods currently available). This allows us to create a large, synthetic state space in a very reasonable timeframe, which can then be used to conduct inference about the parameters.

The method we developed is fast, flexible, highly parallelizable, and robust—the constituent ANNs can be retrained or replaced, individually and as necessary. We achieved a total analysis cost reduction by a factor of 200 after accounting for surrogate model training.

We found that our first attempt at analysis on LVK data pushed our inference to rail against the prior boundaries. In this case, this was likely due to poor choices in the fiducial model describing astrophysical evolution, exacerbated by an overly simplistic observational selection function. We were able to partially remedy this issue because one of the parameters required only a simple rescaling. However, to avoid similar problems in future analyses, it would be wise to use active learning and grow the training data set in regions of parameter space that provide models with the best match to the data.

For this study, we focused on just the four MSSFR parameters used in the post-processing of COMPAS simulations, but we are not limited to those parameters, nor to the post-processing code. The next step is to expand this work to other astrophysical parameters that govern stellar and binary evolution in the population synthesis code itself (e.g., COMPAS), rather than just the post-processing code, thereby reducing the need for time-consuming simulations.

Simulations in this paper made use of the COMPAS rapid population synthesis code, which is freely available at <http://github.com/TeamCOMPAS/COMPAS>. The version of COMPAS used for these simulations was v02.31.00.

Acknowledgments

I.M. is a recipient of the Australian Research Council Future Fellowship FT190100574.

Data availability

The data underlying this article are available from the corresponding author upon request.

Appendix A ANN Training and Performance

We did not use extremely large data sets to train the ANNs that comprise the interpolant in this work. Even so, we achieved very good results with the ANNs we trained—most achieved 100% accuracy when tested on the training and validation data. These results might lead us to wonder whether the ANNs have been overfitted during training (though we did test that they are capable of generalizing).

How many training examples are required to properly train an ANN? The answer depends on the complexity of the problem, as well as the ANN being used to solve it. It is effectively unknowable a priori, and must be discovered through empirical investigation. In other words, we can tell whether the training data were sufficient by testing that the

ANN gets the right answer for the examples it was trained on, and whether it predicts correct answers for inputs it was not trained/validated on. But were the data necessary?

The conventional wisdom is that at least thousands of training examples are required, certainly no fewer than hundreds. Tens, or hundreds, of thousands for an “average” modeling problem; millions, or tens of millions, for a “hard” problem. This basically translates to “Get as much data as possible and use it,” which is not an unreasonable strategy in a situation where a better answer is not immediately apparent.

We know that when a sufficiently large ANN tries to learn from a small data set, it will tend to memorize the entire data set—an example of overfitting. But does this compromise the ability of the network to generalize beyond the training data set? In the past, we might have thought so, but recent work suggests that this is not necessarily the case (Zhang et al. 2017, 2021).

Mukherjee & Huberman (2022) claim that the properties of a particular mathematical operation, expand-and-sparsify (Dasgupta & Tosh 2020), explain the ability of an overparameterized ANN (i.e., the number of parameters in the model exceeds the size of the training data set) to both memorize the entire training data set and generalize beyond the training data set (by learning the underlying structure to the training data).

As noted above, we achieved very good results with the ANNs we trained, yet in most cases the number of epochs over which the networks were trained, and so the time to train, was not particularly onerous. This, especially when coupled with the performance achieved by the ANNs when trained on a relatively small data set, raises the question of why it was not more difficult and time-consuming to find a solution to such a difficult problem. The answer is two-fold.

First, the phenomenon of untrained networks with randomly initialized weights performing surprisingly well has been reported in the past (e.g., Frankle et al. 2020; Zhang et al. 2022). This is often explained by referring to the Lottery Ticket Hypothesis (Frankle & Carbin 2019), which posits that ANNs are really running large lotteries where subnetworks whose weights have serendipitously been randomly initialized to values that produce good results are scattered throughout the network. Thus, in a sufficiently large, untrained, and randomly initialized ANN, a subnetwork with random weights that performs as well as a trained network can be found, and the remainder of the network is ignored.

Next, as noted in Section 2.4.2, we trained the ANNs using the Keras Adam optimizer, which uses an enhanced stochastic gradient descent algorithm. We know that the standard stochastic gradient descent algorithm is usually sufficient to train an ANN—it is not often that other methods of optimization for the network parameters are required. Chizat et al. (2019) note that, often during training, the parameters of an ANN change very little from their initialized values. Mukherjee and Huberman claim that, because the randomly initialized weights very often are “good enough” (as described above), they require hardly any fine tuning for better performance, so an optimizer such as stochastic gradient descent needs to do very little work to train a network to achieve reasonably good performance. Indeed, Mukherjee & Huberman (2022) contend that the training of an ANN is the third-most important component of a successful ANN solution, behind architecture selection and parameter initialization.

Appendix B

COMPAS Configuration Fiducial Values

We list the default COMPAS assumptions for stellar binary evolution and the distributions from which parameters are sampled in Table 4.

Table 4
Initial Values and Default Settings for the COMPAS Fiducial Model

Description and Name	Value/Range	Note
Distribution		
Initial primary mass $m_{1,i}$	$[5, 150] M_{\odot}$	Kroupa (2001) IMF $\propto m_{1,i}^{-\alpha}$ with $\alpha_{\text{IMF}} = 2.3$ for stars above $5 M_{\odot}$
Initial mass ratio $q_i = m_{2,i}/m_{1,i}$	$[0, 1]$	We assume a flat mass ratio distribution $p(q_i) \propto 1$ with $m_{2,i} \geq 0.1 M_{\odot}$
Initial semimajor axis a_i	$[0.01, 1000]$ au	Distributed flat-in-log $p(a_i) \propto 1/a_i$
Initial metallicity Z_i	$[0.0001, 0.03]$	Distributed uniform-in-log
Initial orbital eccentricity e_i	0	All binaries are assumed to be circular at birth
Fiducial Parameter Settings		
Stellar winds for hydrogen-rich stars	Belczynski et al. (2010)	Based on Vink et al. (2000, 2001), including LBV wind mass loss with $f_{\text{LBV}} = 1.5$
Stellar winds for hydrogen-poor helium stars	Belczynski et al. (2010)	Based on Hamann & Koesterke (1998) and Vink & de Koter (2005)
Max transfer stability criteria	ζ -prescription	Based on Vigna-Gómez et al. (2018) and references therein
Mass transfer accretion rate	Thermal timescale	Limited by thermal timescale for stars; see Hurley et al. (2002) and Vinciguerra et al. (2020)
	Eddington-limited	Accretion rate is Eddington-limit for compact objects
Nonconservative mass loss	Isotropic re-emission	Mashevitch & Yungelson (1975); Bhattacharya & van den Heuvel (1991); Soberman et al. (1997) Tauris & van den Heuvel (2006)
Case BB mass transfer stability	Always stable	Based on Tauris et al. (2015, 2017) and Vigna-Gómez et al. (2018)
CE prescription	$\alpha - \lambda$	Based on Webbink (1984) and De Kool (1990)
CE efficiency α -parameter	1.0	
CE λ -parameter	λ_{Nanjing}	Based on Xu & Li (2010a, 2010b) and Dominik et al. (2012)
Hertzsprung gap (HG) donor in CE	Pessimistic	Defined in Dominik et al. (2012): HG donors do not survive a CE phase
SN natal kick magnitude v_k	$[0, \infty)$ km s ⁻¹	Drawn from Maxwellian distribution with standard deviation $\sigma_{\text{rms}}^{\text{ID}}$
SN natal kick polar angle θ_k	$[0, \pi]$	$p(\theta_k) = \sin(\theta_k)/2$
SN natal kick azimuthal angle ϕ_k	$[0, 2\pi]$	Uniform $p(\phi) = 1/(2\pi)$
SN mean anomaly of the orbit	$[0, 2\pi]$	Uniformly distributed
Core-collapse SN remnant mass prescription	Delayed	From Fryer et al. (2012), which has no lower BH mass gap
USSN remnant mass prescription	Delayed	From Fryer et al. (2012)
ECSN remnant mass prescription	$m_f = 1.26 M_{\odot}$	Based on Equation (8) in Timmes et al. (1996)
Core-collapse SN velocity dispersion $\sigma_{\text{rms}}^{\text{ID}}$	265 km s ⁻¹	1D rms value based on Hobbs et al. (2005)
USSN and ECSN velocity dispersion $\sigma_{\text{rms}}^{\text{ID}}$	30 km s ⁻¹	1D rms value based on, e.g., Pfahl et al. (2002) and Podsiadlowski et al. (2004)
PISN / PPISN remnant mass prescription	Marchant et al. (2019)	As implemented in Stevenson et al. (2019)
Maximum NS mass	$\text{max}_{\text{NS}} = 2.5 M_{\odot}$	Following Fryer et al. (2012)
Tides and rotation		We do not include tides and/or rotation in this study
Binary fraction	$f_{\text{bin}} = 0.7$	
Solar metallicity Z_{\odot}	$Z_{\odot} = 0.0142$	Based on Asplund et al. (2009)
Simulation Settings		
Binary population synthesis code	COMPAS	Stevenson et al. (2017); Barrett et al. (2018); Vigna-Gómez et al. (2018); Neijssel et al. (2019) Broekgaarden et al. (2019); Team COMPAS et al. (2022a).

ORCID iDs

Jeff Riley  <https://orcid.org/0000-0003-1530-2557>Ilya Mandel  <https://orcid.org/0000-0002-6134-8946>

References

- Abadi, M., Agarwal, A., Barham, P., et al. 2015, arXiv:1603.04467
- Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2020, *LRR*, **23**, 3
- Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. 2003, *Mach. Learn.*, **50**, 5
- Asplund, M., Grevesse, N., Sauval, A. J., & Scott, P. 2009, *ARA&A*, **47**, 481
- Barrett, J. W., Gaebel, S. M., Neijssel, C. J., et al. 2018, *MNRAS*, **477**, 4685
- Barrett, J. W., Mandel, I., Neijssel, C. J., Stevenson, S., & Vigna-Gómez, A. 2016, in IAU Symp. 325, *Astrophysics* (Cambridge: Cambridge Univ. Press), 46
- Belczynski, K., Bulik, T., Fryer, C. L., et al. 2010, *ApJ*, **714**, 1217
- Belczynski, K., Kalogera, V., Rasio, F. A., et al. 2008, *ApJS*, **174**, 223
- Bhattacharya, D., & van den Heuvel, E. P. J. 1991, *PhR*, **203**, 1
- Breivik, K., Coughlin, S., Zevin, M., et al. 2020, *ApJ*, **898**, 71
- Broekgaarden, F. S., Berger, E., Stevenson, S., et al. 2022, *MNRAS*, **516**, 5737
- Broekgaarden, F. S., Justham, S., de Mink, S. E., et al. 2019, *MNRAS*, **490**, 5228
- Cheung, D. H. T., Wong, K. W. K., Hannuksela, O. A., Li, T. G. F., & Ho, S. 2022, *PhRvD*, **106**, 083014
- Chizat, L., Oyallon, E., & Bach, F. 2019, in 33rd Conf. on Neural Information Processing Systems, ed. H. Wallach, H. Larochelle, A. Beygelzimer et al. (Red Hook, NY: Curran Associates Inc.), 2937
- Chollet, F., et al. 2015, Keras, <https://keras.io>
- Dasgupta, S., & Tosh, C. 2020, arXiv:2006.03741
- De Kool, M. 1990, *ApJ*, **358**, 189
- Dominik, M., Belczynski, K., Fryer, C., et al. 2012, *ApJ*, **759**, 52
- Farr, W. M., Gair, J. R., Mandel, I., & Cutler, C. 2015, *PhRvD*, **91**, 023005
- Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, *PASP*, **125**, 306
- Frankle, J., & Carbin, M. 2019, in 7th Int. Conf. on Learning Representations, <https://openreview.net/forum?id=rJl-b3RcF7>
- Frankle, J., Schwab, D. J., & Morcos, A. S. 2020, arXiv:2003.00152
- Fryer, C. L., Belczynski, K., Wiktorowicz, G., et al. 2012, *ApJ*, **749**, 91
- Gerosa, D., & Fishbach, M. 2021, *NatAs*, **5**, 749
- Hamann, W. R., & Koesterke, L. 1998, *A&A*, **335**, 1003
- Heger, A., & Woosley, S. E. 2002, *ApJ*, **567**, 532
- Hobbs, G., Lorimer, D. R., Lyne, A. G., & Kramer, M. 2005, *MNRAS*, **360**, 974
- Hurley, J. R., Tout, C. A., & Pols, O. R. 2002, *MNRAS*, **329**, 897
- Iorio, G., Costa, G., Mapelli, M., et al. 2022, arXiv:2211.11774
- Izzard, R. G., Preece, H., Jofre, P., et al. 2018, *MNRAS*, **473**, 2984
- Kroupa, P. 2001, *MNRAS*, **322**, 231
- Langer, N., & Norman, C. A. 2006, *ApJ*, **638**, L63
- LIGO Scientific Collaboration, Virgo Collaboration, KAGRA Collaboration 2021, GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo During the Second Part of the Third Observing Run—Parameter estimation data release v1, Zenodo, doi:10.5281/zenodo.5546663
- LIGO Scientific Collaboration, & Parameter Virgo Collaborations 2022, 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 v2, Zenodo, doi:10.5281/zenodo.6513631
- Lin, L., Bingham, D., Broekgaarden, F., & Mandel, I. 2021, *AnApS*, **15**, 1604
- Madau, P., & Dickinson, M. 2014, *ARA&A*, **52**, 415
- Mandel, I., & Farmer, A. 2022, *PhR*, **955**, 1
- Mandel, I., Farr, W. M., & Gair, J. R. 2019, *MNRAS*, **486**, 1086
- Marchant, P., Langer, N., Podsiadlowski, P., Tauris, T. M., & Moriya, T. J. 2016, *A&A*, **588**, A50
- Marchant, P., & Moriya, T. J. 2020, *A&A*, **640**, L18
- Marchant, P., Renzo, M., Farmer, R., et al. 2019, *ApJ*, **882**, 36
- Massevitch, A., & Yungelson, L. 1975, *MmSAI*, **46**, 217
- Mukherjee, S., & Huberman, B. A. 2022, arXiv:2211.14632
- Neijssel, C. J., Vigna-Gómez, A., Stevenson, S., et al. 2019, *MNRAS*, **490**, 3740
- Pfahl, E., Rappaport, S., & Podsiadlowski, P. 2002, *ApJ*, **573**, 283
- Podsiadlowski, P., Langer, N., Poelarens, A. J. T., et al. 2004, *ApJ*, **612**, 1044
- Powell, J., Stevenson, S., Mandel, I., & TiÅo, P. 2019, *MNRAS*, **488**, 3810
- Rasmussen, C. E., & Williams, C. K. I. 2005, *Gaussian Processes for Machine Learning* (Cambridge, MA: MIT Press)
- Riley, J., Mandel, I., Marchant, P., et al. 2021, *MNRAS*, **505**, 663
- Russell, S. J., & Norvig, P. 1995, *Artificial Intelligence: A Modern Approach* (Englewood Cliffs, NJ: Prentice Hall), 111
- Soberman, G. E., Phinney, E. S., & van den Heuvel, E. P. J. 1997, *A&A*, **327**, 620
- Stevenson, S., Sampson, M., Powell, J., et al. 2019, *ApJ*, **882**, 121
- Stevenson, S., Vigna-Gómez, A., Mandel, I., et al. 2017, *NatCo*, **8**, 14906
- Tagawa, H., Kocsis, B., Haiman, Z., et al. 2021, *ApJ*, **908**, 194
- Tauris, T. M., Kramer, M., Freire, P. C. C., et al. 2017, *ApJ*, **846**, 170
- Tauris, T. M., Langer, N., & Podsiadlowski, P. 2015, *MNRAS*, **451**, 2123
- Tauris, T. M., & van den Heuvel, E. P. J. 2006, *Compact Stellar X-ray Sources*, Vol. 39 (Cambridge: Cambridge Univ. Press), 623
- Taylor, S. R., & Gerosa, D. 2018, *PhRvD*, **98**, 083017
- Team COMPAS, Riley, J., Agrawal, P., et al. 2022a, *ApJS*, **258**, 34
- Team COMPAS, Riley, J., Agrawal, P., et al. 2022b, *JOSS*, **7**, 3838
- Timmer, F. X., Woosley, S. E., & Weaver, T. A. 1996, *ApJ*, **457**, 834
- Van Rossum, G., & Drake, F. L. 2009, *Python 3 Reference Manual* (Scotts Valley, CA: CreateSpace)
- van Son, L. A. C., de Mink, S. E., Chruslinska, M., et al. 2022, arXiv:2209.03385
- Vigna-Gómez, A., Neijssel, C. J., Stevenson, S., et al. 2018, *MNRAS*, **481**, 4009
- Vinciguerra, S., Neijssel, C. J., Vigna-Gómez, A., et al. 2020, *MNRAS*, **498**, 4705
- Vink, J. S., & de Koter, A. 2005, *A&A*, **442**, 587
- Vink, J. S., & Koter, A., & Lamers, H. J. G. L. M. 2000, *A&A*, **362**, 295
- Vink, J. S., de Koter, A., & Lamers, H. J. G. L. M. 2001, *A&A*, **369**, 574
- Webbink, R. F. 1984, *ApJ*, **277**, 355
- Wong, K. W. K., Breivik, K., Kremer, K., & Callister, T. 2021, *PhRvD*, **103**, 083021
- Wong, K. W. K., & Gerosa, D. 2019, *PhRvD*, **100**, 083015
- Xu, X.-J., & Li, X.-D. 2010a, *ApJ*, **716**, 114
- Xu, X.-J., & Li, X.-D. 2010b, *ApJ*, **722**, 1985
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. 2017, in 5th Int. Conf. on Learning Representations, <https://openreview.net/forum?id=Sy8gdB9xx>
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. 2021, *Commun. ACM*, **64**, 107
- Zhang, C., Bengio, S., & Singer, Y. 2022, *JMetr*, **23**, 1