Identifying and mitigating machine-learning biases for the gravitational-wave detection problem

Narenraju Nagarajan[*] and Christopher Messenger[†]

SUPA, School of Physics and Astronomy, University of Glasgow, Glasgow G12 8QQ, United Kingdom

(Received 27 January 2025; accepted 3 October 2025; published 4 November 2025)

Matched filtering is a long-standing technique for the optimal detection of known signals in stationary Gaussian noise. However, it has known departures from optimality when operating on unknown signals in real noise and suffers from computational inefficiencies in its pursuit of near-optimality. A compelling alternative that has emerged in recent years to address this problem is deep learning. Although it has shown significant promise when applied to the search for gravitational-waves in detector noise, we demonstrate the existence of a multitude of learning biases that hinder generalization and lead to significant loss in detection sensitivity. Our work identifies the sources of a set of 11 interconnected biases present in the supervised learning of the gravitational-wave detection problem and contributes mitigation tactics and training strategies to concurrently address them. In light of the identified biases, we demonstrate that existing detection sensitivity metrics are not reliable for machine-learning pipelines and discuss the trustworthiness of previous results. We use gravitational-wave domain knowledge to build a bespoke machine-learning-based binary black hole search pipeline called Sage that addresses these biases. Via the injection study presented in the Machine-Learning Gravitational-Wave Search Challenge, we show that Sage detects ≈11.2% more signals than the benchmark PyCBC analysis at a false-alarm rate of one per month in O3a noise. Moreover, we also show that it can detect ≈48.29% more signals than the previous best-performing machine-learning pipeline on the same dataset. We empirically prove that Sage can (i) effectively handle out-of-distribution noise power spectral densities, (ii) strongly reject non-Gaussian transient noise artifacts, and (iii) achieve higher detection sensitivities using less data than network architectures of a similar size. By studying machine-learning biases and conducting empirical investigations to understand the reasons for performance improvement or degradation, we aim to address the need for interpretability of machine-learning methods for gravitational-wave science. All code and implementations are available.

DOI: 10.1103/zwj9-ycyz

I. INTRODUCTION

The promise of deep learning in the field of applied data science has been to develop fast approximation machines capable of emulating physical principles. Although it has been successful in seemingly approaching this ideal, it falls short due to its inability to truly generalize or adapt outside its training distribution (otherwise called brittleness) [1,2], being biased [3] and data intensive [4]. We hope that, by acknowledging the issues with machine-learning (ML), we can be better informed of their capabilities and, more importantly, their deficits. While these deficits exist, it is possible to mitigate them and produce reliable pipelines via careful consideration of domain-specific knowledge. In this paper, we focus on identifying and addressing some of the biases that prevail in the application of ML to the gravitational-wave (GW) detection of binary black-hole (BBH) mergers. Although we focus on the BBH detection problem and provide explanations from this perspective, the salient ideas can be extrapolated and adapted to suit other sources of GWs and domains of GW data analysis.

Matched filtering [5–8] is an optimal linear filter that maximizes the signal-to-noise ratio (SNR) in additive white Gaussian noise for a known signal [9]. With careful considerations for the detection of unknown GW signals in colored Gaussian noise and non-Gaussian transient noise artifacts [10], it has aided in the discovery of 90 GW events as of the third observation run of the advanced GW detector network [11–20]. To better understand the necessity for looking at alternate ways of detecting GWs, we first build a case against matched filtering in Sec. I A. We then highlight the advantages that an alternate ML-based method has over

[*]Contact author: n.nagarajan.1@research.gla.ac.uk
[†]Contact author: christopher.messenger@glasgow.ac.uk

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

matched filtering in Sec. I B but also consider its current issues in Sec. I C. We then provide effective mitigation tactics, in Sec. II, that aim to address these issues concurrently. To validate our methods, we conduct an injection study with 1 month of O3a noise from the Hanford (H1) and Livingston (L1) detectors (see Sec. III), as described in the Machine-Learning Gravitational-Wave Search Mock Data Challenge (MLGWSC-1) [21]. This includes a comparative investigation of the detection performance of our pipeline, Sage, with the production-level matched-filtering pipeline, PyCBC [8,22–24], and previous works in ML-based GW detection. Finally, we systematically remove important components of Sage and investigate their contribution to the overall detection performance, thus justifying our method choices in Sec. II.

Although investigations into these biases have existed in the machine-learning literature in some form, we translate it to the GW detection problem and provide methods to mitigate them. We show that addressing these biases leads to a significant boost in detection performance compared to previous works and argue that the simultaneous mitigation of these issues is a necessary condition for ML to match matched filtering. As a consequence, we show that Sage can detect more signals than matched filtering while operating on the same parameter space and is capable of effective glitch mitigation.

A. Matched filtering for GW detection

The definition of an optimal receiver is often muddied when dealing with real-world data [9], such as for the detection of unknown GWs in detector noise, making the matched-filter SNR a nonoptimal detection statistic. We can deconstruct the major deviations from the optimal detection scenario into the following: unknown signal parameters, nonstationary noise, uncertainty in the estimates of noise power spectral density (PSD), and non-Gaussian noise artifacts. All of these factors contribute to the departure of matched filtering from Neyman-Pearson optimality [25–29] to varying degrees [9,10].

Say we have the time series data $d = n + h$ collected from a GW detector, where n is the combination of all possible noise sources and h is the sum of the GW response of the detector. In the case of an unknown signal, the optimal detection statistic would be obtained by marginalizing the likelihood ratio $\Lambda(d|\theta)$ over all unknown signal parameters, θ, by integrating $\Lambda(d|\theta)$ over θ. The likelihood ratio will generally be sharply peaked about its maximum, given that the log-likelihood ratio is a linear function of the signal model. Thus, maximizing $\Lambda(d|\theta)$ over θ is expected to be a good approximation of the marginalized likelihood ratio, up to a constant rescaling factor [10]. However, this approximation is a nonoptimal treatment of the parameter space.

The template bank used for matched filtering over the multidimensional GW signal parameter space is constructed such that at least one template in the bank is close enough to a potential astrophysical signal while not losing more than a predetermined fraction of the SNR under ideal conditions [5,6,30]. Although this condition leads to a reasonably sized template bank for compact binary coalescences (CBC) in the current detector era, it is well known that template-bank-based approaches will become too computationally expensive in the future [31,32].

Deviations of the detector noise from independent and identically distributed (IID) Gaussian noise, in the time domain, include colored noise, nonstationary noise, and the presence of non-Gaussian transient instrumental artifacts arising from known and unknown sources [33–35]. For a known noise covariance matrix, we can reduce the problem of detecting a signal in colored noise to the problem of detecting a signal in colored noise to the problem of detecting a signal in IID Gaussian noise. In practice, this is typically performed by whitening the signal in the frequency domain by estimating the PSD using the off-source or on-source methods. Both methods assume that the estimated PSD is exact and that the noise is stationary and Gaussian. These assumptions do not hold in either case, but more so for the off-source method, since it requires a longer duration of data than the on-source method [36].

The presence of non-Gaussian transient instrumental artifacts or glitches can induce certain templates to produce high-SNR outputs, making the matched-filter SNR highly nonoptimal as a detection statistic [10]. This is currently mitigated using witness sensors [37], waveform consistency checks [8,22,38,39], and time coincidence tests [40]. However, artifacts of unknown origin that are similar in time and frequency evolution to GW signals are difficult to veto out. This is especially true in the case of short-duration GW signals, where the background might be contaminated by numerous short-duration transient noise [41]. Although there exist waveform consistency tests to identify these types of noise, they are based on assumptions that might not always hold true and would thereby target only a small subset of all possible glitches [42]. These assumptions include modeling short-duration noise transients as sine-Gaussian, targeting a particular frequency range for discriminative features, and using parameter distributions for the assumed noise model that may not completely coincide with the truth [42]. Such assumptions have to be made, since it is impossible to accurately model noise of unknown origin or create a waveform consistency test that is reliable for all future observed glitches. The utmost that we can do to mitigate the effects of unknown noise on detection efficiency is to incorporate all prior non-Gaussian noise data into a generalized waveform consistency test that is capable of being continually updated. Deep learning is a promising framework that is capable of handling problems of this nature.

B. Machine-learning for GW detection

The subtle but fundamental difference that allows for the increased efficiency of machine-learning-based searches when compared to matched filtering is the use of a nonparametric hypothesis instead of its parametric counterpart—"is this a GW?" instead of "is this a GW with parameters θ?" A nonparametric approach allows for a template-free search throughout the considered parameter space.

Problem setup. The search for GWs, in the context of supervised machine-learning, can be interpreted as a binary classification problem between a pure noise and noisy signal class. Typically, we are given a collection of n training data pairs, $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$, where $\mathbf{x}_i$ is some input data sample and $\mathbf{y}_i$ is its corresponding class label. In our case, the noise class contains detector noise taken from the third advanced detector observing run (O3) [11]. The signal class contains simulated GW signals with different realizations of additive detector noise. The signals are generated by randomizing on the standard parameter distributions of all intrinsic and extrinsic GW parameters. We seek an ML *model* Φ with two primary properties. First, the model should be able to *interpolate* the training data such that $\Phi(\mathbf{x}_i) \approx \mathbf{y}_i$. Second, the model should learn a *generalized* representation of the training data, such that, when it is shown an unseen testing data pair $(\mathbf{x}_i', \mathbf{y}_i')$, $\Phi(\mathbf{x}_i') \approx \mathbf{y}_i'$. If the model has interpolated the training data but does not generalize, the model is said to have *overfit*. The output of the model $\Phi(\mathbf{x}_i')$ can be interpreted as a ranking statistic of the provided input sample $\mathbf{x}_i'$, and if $\Phi(\mathbf{x}_i')$ is normalized to lie in [0, 1], it can be interpreted as a prediction probability. During the *training* or *learning* phase, the trainable parameters Θ of the model Φ is optimized to minimize a *loss function* that reduces some distance measure between $\Phi(\mathbf{x}_i)$ and $\mathbf{y}_i$ [43].

Machine-learning is a promising alternative to tackle the detection problem in nonoptimal settings. This is corroborated by previous machine-learning literature, some of which are chosen here: [44,45] exhibit the capability of ML in detecting low-SNR signals, [46] classifies accurately on highly complex and large datasets, and [47,48] provide theoretical insights into why ML generalizes well. It has also been shown that neural networks have the capability to approach Neyman-Pearson optimality for the noisy signal detection problem when trained with a sufficient amount of data on a large enough network under appropriate training strategies [49,50]. Thus, the ML literature has more than sufficient evidence for its capability to handle GWs in detector noise when used appropriately.

The usage of ML for GW detection is strongly supported, given that GWs from BBH mergers were successfully distinguished from noise samples using machine-learning, as shown in the progenitor works [51,52]. The Neyman-Pearson optimality of these search methods was later proven in [29]. Although these studies have demonstrated that they match or surpass matched filtering, it is albeit under restricted conditions. Under more realistic conditions, [21] showed that machine-learning achieves at most ≈70% of the detection sensitivity when compared to PyCBC at a false-alarm rate (FAR) of one per month in O3a noise. AresGW, one of the teams that participated in the mock data challenge [21], later showed that they were able to achieve detection sensitivities higher than PyCBC at all false-alarm rates above one per month [53].[1] At face value, achieving a higher sensitive distance, as defined in [21], than PyCBC suggests that machine-learning has surpassed matched filtering at the GW detection problem. In the next section, we will argue against this statement and discuss the necessary conditions that need to be satisfied for ML to match matched filtering in detection performance.

C. Acknowledging potential problems of machine-learning models

In this section, we discuss a comprehensive set of issues that systematically bias ML models to disproportionately favor the learning of certain parts of the input parameter space more than others. Say we have two GW signals with parameters θ and θ' in some noise realization. There is a maximum amount of *information* that can be extracted from these noisy signals, for the purpose of classification. The supervised training of an ML model on these noisy signals is not guaranteed to extract the maximum possible information for every possible GW signal. If the amount of information learned from a GW signal with parameters θ differs from that learned from a signal with parameters θ', then the ML model is said to exhibit a learning bias. In this section, we identify and describe several possible sources of bias that could affect the effective learning of the GW parameter space.

It is important to note that the issues discussed in this section correspond to ML models trained using a *supervised approach* on a labeled dataset, as typically seen in previous works [21,29,51–53,55,56]. These issues have generally been well documented in the machine-learning literature but have not become standard in the context of GW detection using ML. A subset of the following issues were discussed in [57], where they show different modes of failure for ML that hinder detection performance. In this work, we try to provide a more comprehensive set of issues and investigate effective mitigation strategies that should serve as a guide to developing better ML pipelines.

1. Definitions

Let us consider the inner product $(.|.)$ [58] between two real functions a and b expressed in the frequency domain as

[1]The AresGW pipeline was later enhanced in [54]. We discuss more about this in Sec. III B. Any mention of AresGW throughout this paper refers to [53], and the term enhanced-AresGW refers to [54].

$$(a|b) = 2\int_0^\infty \frac{\tilde{a}(f)\tilde{b}^*(f) + \tilde{a}^*(f)\tilde{b}(f)}{S(f)}\,df, \quad (1)$$

where $S(f)$ is the one-sided power spectral density and $*$ represents the complex conjugate operation. The optimal SNR is defined as the square root of the norm of h:

$$(h|h) = 4\int_0^\infty \frac{|\tilde{h}(f)|^2}{S(f)}\,df, \quad (2)$$

where h is the Fourier-transformed waveform. The matched-filter SNR ρ is defined as

$$\rho = \frac{(s|h)}{\sqrt{(h|h)}}, \quad (3)$$

where s is the Fourier-transformed detector data. The term integrated SNR is used interchangeably with the matched-filter SNR within the text.

Inductive bias. Almost all learning biases can be directly or indirectly related to *inductive bias*. Inductive bias is the biased preference in machine-learning models to constrain the hypothesis space, given the training data. A simple example is the difference between a 100th order polynomial fit and a 100-parameter neural network trying to fit discrete data points generated by a fifth-order polynomial. The polynomial fit will have large errors, but the over-parametrized neural network is capable of fitting the relatively simple function easily. This is due to the innate bias of ML models toward learning simpler functions and hypothesis [59,60]. The architecture of the model can be altered to point the inductive bias toward the correct hypotheses using domain knowledge. For example, the detection problem of a set of objects in an image should not depend on the order or arrangement of the objects in that image, so one can use a 2D permutation-invariant architecture that treats all arrangements equally [61]. Unwanted learning biases typically arise from a lack of or improper use of inductive bias.

2. Biases in the training data

Bias due to insufficient information. Insufficient information can be interpreted in a multitude of ways. Say we generate a training dataset using some chirp mass distribution for our ML model. This will determine the chirp times (also referred to as signal duration and defined as the time spent within the detection band prior to the merger) of the signals up to the first order, with lower chirp masses generating signals with longer chirp times. We define the chirp time as a function of the signal low-frequency cutoff f_l as

$$\tau_0(f_l) = t_c - 5(8\pi f_l)^{-8/3}\mathcal{M}_c^{-5/3}, \quad (4)$$

where t_c is the time of coalescence and $\mathcal{M}_c = \frac{(m_1 m_2)^{\frac{3}{5}}}{(m_1+m_2)^{\frac{1}{5}}}$ is the chirp mass [58]. The length of the samples used for the training dataset must be long enough to contain most of the integrated SNR of all possible signal durations dictated by the chirp mass distribution. Using an insufficient sample length leads to reduced integrated SNR for signals that are longer than that sample length in the training dataset. This biases the ML model, proportionate to the amount of lost SNR, against low chirp mass signals. Some previous works, such as the ML pipelines in MLGWSC-1 [21], AresGW [53], and Aframe [56], are affected by this bias. This bias negates the need for the ML model to learn discriminative features from lower chirp mass signals and narrows the operating parameter space. This situation is similar to running a matched-filter search on a narrower parameter space. Assuming Gaussian noise, the false-alarm threshold on a detection statistic at a given false-alarm rate will be lowered compared to a matched-filter search on the full parameter space. This allows AresGW to discover more high chirp mass signals than PyCBC that is running on the entire parameter space, at a given FAR, in the mock data challenge dataset [53].

Other examples include using an insufficient sampling frequency to represent merger and ringdown or using a filter to discard important discriminative features present in non-Gaussian noise transients. Although removing glitch features might seem intuitively correct, we found that keeping them within the training dataset aided in rejecting unseen glitches and allowed the ML model to be more confident in separating noise from signals. We will explore this further in Sec. III A 4.

Bias due to class overlap. The boundary between a detectable GW and pure noise is soft and broad. In the context of generating a set of noisy signals for the training data, we have freedom over the choice of optimal SNR. When injected into arbitrary samples of detector noise, the resulting distribution of matched-filter SNR will have an innate variance around the optimal SNR. Moreover, in non-Gaussian noise, the matched-filter SNR threshold above which a trigger can be considered an event at a reasonable FAR is not well defined and can vary within the parameter space. Our choice of training data GW parameter distributions, alongside the considered detector noise realization, dictates the extent of overlap between the signal and noise classes. This bias also ties back to the *bias due to insufficient information*—all signals with an integrated SNR lower than the minimum SNR threshold are essentially noise and are, thus, wrongly labeled. Wrong or noisy labeling in the class overlap problem is associated with brittle generalization [62,63].

Bias due to imbalance between classes. Supervised machine-learning is known to learn the distribution of the data, including any innate biases between classes, regardless of the relative importance of each sample [3].

Counteracting the noise dominance of the training dataset by using more samples in the signal class compared to the noise class has been seen in previous works [57,64]. Introducing a bias toward the signal class will lead to worse noise rejection capabilities at lower FARs. Since reducing the loss function amounts only to placing more samples closer to their corresponding class label, the imbalance leads to an increase in false positives for the sake of increasing true positives. The joint effect of class overlap and imbalance is still poorly understood, but it is widely agreed that both phenomena impede unbiased model learning.

Bias due to lack of variation. Variation within the signal class is defined by the changes in morphology produced by differences in the intrinsic and extrinsic GW parameters. It is typical in supervised learning settings to generate a limited training dataset and to augment the samples to introduce more variation. This is required across dimensions in the parameter space, along which the classifier cannot interpolate trivially. The inability to interpolate trivially is defined as the inability of the model to comprehend a continuous transformation space from the given finite set of discrete transformed instances. It is difficult to determine the number of discrete transformations or its bounds in the continuous space, required for a given model to understand the overall transformation. This will vary depending on the GW parameter in question. For example, although a simple scaling transformation as seen in SNR can be understood easily with a relatively small number of discrete SNR instances within a sufficient upper and lower bound, the same might not be true for chirp mass or mass ratio, which require more complex transformations.

Similarly, for the noise class, variation amounts to different estimated PSDs, different noise realizations, and the presence of distinct glitches. Assuming that we use real detector noise for the training dataset, it is safe to make some assumptions. First, the different noise realizations are uncorrelated. Second, the different instances within a glitch type are not reliably present within a continuous transformation space. One of the easiest ways to perform good noise rejection is to provide the model with more training data. However, due to limitations in the amount of available detector noise, there is a possibility for the network to become biased toward the available data.

Bias due to limited sample representation. The choice of the training distribution on intrinsic and extrinsic GW parameters is one of the primary sources of biased learning. Say we sample uniform in the $(m_1, m_2) = U[7, 50]M_\odot$ space subject to the constraint $m_1 > m_2$, where m_1 and m_2 are the binary source masses. Although this is a very common choice and a uniform sampling for (m_1, m_2) might seem akin to an unbiased representation, evidence of the contrary can be appreciated by considering the associated chirp time τ_0. Figure 1 shows lines of constant τ_0 in the (m_1, m_2) space. ≈25.1% of the samples are above a duration of 2 s, ≈4.6% above 4 s and only ≈0.22% above 8 s. Given that the longest τ_0, obtained via Eq. (4) for this parameter space, should be around 11 s, a training dataset built in this framework will lead to heavily biased learning against longer τ_0. Using an input sample length that is longer than the longest-duration signal (addressing *bias due to insufficient information*) is not solely sufficient to claim to have built a model that is operating equally on the full parameter space. This bias can also be related to the *bias due to lack of sufficient variation* in the τ_0 dimension. A change in τ_0, with all other parameters fixed, is a linear time dilation or contraction transformation. Even if we had a distribution that is uniform on τ_0, the difficulty of learning different signal durations is directly proportional to τ_0 (see *bias due to sample difficulty*), meaning uniform on τ_0 is still not sufficient to be unbiased.

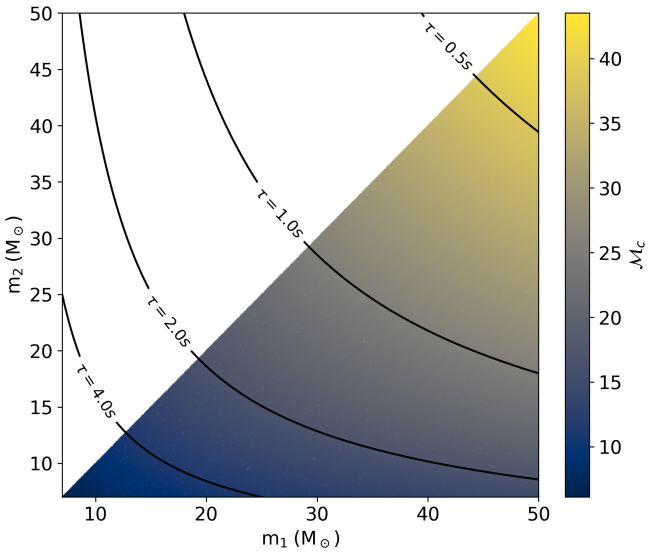


FIG. 1. Diagram showing the boundaries of the (m_1, m_2) space colored by chirp mass $\mathcal{M}_c$. The solid lines show curves of constant chirp time calculated using Eq. (4).

Previous works like [21,54,56] acknowledged the existence of this bias for chirp mass due to lower detection sensitivities observed for low chirp mass signals.

Bias due to limited feature representation. A feature is defined as an individual measurable property or attribute of data in ML and pattern recognition according to [65]. In our problem, this can include frequency, amplitude, and phase evolution profiles of GWs, properties of noise PSDs and glitches, possible correlations between detector data in a coherent search, or a lack of discriminative features in white Gaussian noise.

Say we choose a parameter distribution for the training dataset that is uniform on τ_0. We could define a feature in the frequency evolution profile of the GWs to be the instantaneous frequency and SNR pair along this profile. The instantaneous SNR of BBH chirps will typically be higher at higher frequencies till merger. The ML model

could, thus, disproportionately place a lower weight toward learning early inspiral features.

Similarly, with detector noise, there could be realizations of PSD and glitches, with outlier features that are underrepresented in the training data. A lower weight toward learning these realizations would lead to the model not being able to confidently reject similar realizations of noise in the unseen testing dataset, leading to worse performance at lower FARs.

Bias due to train-test mismatch. The typical desire in machine-learning is consistency between training and testing feature distributions, and a mismatch might lead to degraded generalization and overly optimistic measures of performance during training [66]. In the context of GW detection, this is particularly important, as models trained on idealized injections may underperform when exposed to real detector noise with unmodeled nonstationarities.

3. Biases in the learning model

Bias due to sample difficulty. Not all samples are equally learnable for a given ML model if the proper inductive biases are not incorporated into its architecture [67]. Learning long-term dependencies has been a persistent issue in ML [68], and many architectural advances have been made to tackle it [69–71]. Solving this issue hinges on two factors: the largest region of the input sample that the model can use to create a feature, also known as the *maximum effective receptive field* [72], and the ability to treat dependencies over different distances (near-)equally. For our problem, both factors depend on proper inductive biases of the architecture, and the former also depends on the maximum extent to which the input can be compressed. If the receptive field is not large enough to handle the longest possible signal in the input data, providing the model with 100% of the SNR (addressing *bias due to insufficient information*) is not sufficient to ensure unbiased learning. Since standard ML architectures must be deeper and take longer to learn long-term dependencies [72] and this is the hallmark of example difficulty for a given architecture according to [67], we can conclude that longer signals are more difficult to learn. Even with a sufficient receptive field and 100% of the SNR, a standard architecture will be biased against longer-duration signals due to relative example difficulty.

Spectral bias. Work on spectral bias [73,74] has shown that neural networks are biased toward learning lower-frequency features first, and their generalization capability degrades with an increase in the frequency content of the features. In the context of GW chirps, this amounts to generalizing differently to different GW frequency profiles and even parts of a given chirp signal. The bias due to the skewed concentration of integrated SNR toward higher frequencies will amalgamate with spectral bias and result in some uncertain learning dynamics. With regard to detector noise, neural networks should be more prone to higher-frequency noise than lower-frequency. Models may struggle to consistently differentiate between sharp glitch features and the late inspiral or merger phases of true GW signals, potentially leading to degraded detection sensitivity or misclassification in these regions.

4. Experimenter bias

Bias due to disproportionate evaluation. The mock data challenge, MLGWSC-1 [21], describes the sensitive volume as

$$\mathcal{V}(\mathcal{F}) \approx \frac{V(d_{\max})}{N_I} \sum_{i=1}^{N_{I,\mathcal{F}}} \left(\frac{\mathcal{M}_{c,i}}{\mathcal{M}_{c,\max}} \right)^{5/2}, \tag{5}$$

where $V(d_{\max})$ is the volume enclosed within a radius $d_{\max}$, $\mathcal{F}$ is the FAR, $d_{\max}$ is the maximum luminosity distance of all injected sources, N_I is the number of simulated signals present in the data, $N_{I,\mathcal{F}}$ is the number of found signals at a given FAR, and $\mathcal{M}_c$ is the chirp mass. The sensitive volume metric is evaluated at the required false-alarm rates, and the sensitive distance is defined as $(3 \times \mathcal{V}(\mathcal{F})/4\pi)^{1/3}$. The metric places an increased weighting to higher chirp mass detections given by $(\mathcal{M}_{c,i}/\mathcal{M}_{c,\max})^{5/2}$ on each detection indexed by i. An ML model that is biased against low chirp mass signals and operating on a subset of the parameter space considered by matched filtering will find more high chirp mass signals comparatively at a given FAR. Given the imbalanced chirp mass weighting, the biased model will achieve a higher sensitive distance but detect a lower number of signals overall compared to matched filtering at all FARs.

Bias due to train-test overlap. Train-test overlap is measured as the extent to which the testing data features are present in the training data. Significant overlap will lead to overly optimistic evaluation results and will not reflect the true generalization performance of the model on unseen data.

For the signal class, the features are defined in a continuous parameter space with well-defined bounds given by the chosen training distributions, which are expected to be the same between the training and testing datasets. In this case, we seek an ML model that interpolates the training data. Detrimental overlap for the signal class amounts to having signals produced using the same combination of GW parameters in both datasets. Signs of model overfitting will be masked due to this overlap. However, approximate overlap, where the parameters are similar but not the same (from the same distribution), will always exist between the datasets and is not detrimental.

For the noise class, no well-defined definition exists for the features, and there is often a mismatch between training and testing datasets. In this case, we seek an ML model that *extrapolates* the training data. Detrimental overlap for the noise class amounts to using the same realizations of noise

PSD or glitches in both datasets. Using a testing dataset that does not have appropriate out-of-distribution (OOD) noise realizations is detrimental, as it does not reflect the model's extrapolation capabilities.

D. Key contributions to the field

Identified sources of learning bias. We provide explanations for several sources of ML learning bias for the BBH GW detection problem. However, they are general enough to be easily extrapolated to other domains of GW data analysis.

Provided bias mitigation tactics. In Sec. II, we will provide translatable mitigation tactics and training strategies to *concurrently* address almost all the aforementioned biases.

Improvements to detection performance. In Sec. III A, we evaluate our pipeline on the injection study presented in MLGWSC-1 and show that our detection pipeline, Sage, is capable of detecting ≈11.2% and ≈48.29% more signals than the PyCBC submission in [21] and the AresGW results in [53], at an FAR of one per month in O3a noise.

Negating the need for PSD estimation. Contrary to previous ML-based detection pipelines operating on real detector noise [53,54,56,64], we show that Sage does not need to use a PSD estimate for whitening any input sample (see Sec. II E). We use a single approximate PSD instead for all samples. The primary benefit of this feature is not that it reduces computational cost during data preprocessing, but the idea of having "less moving parts" in a low-latency search setting, thus reducing the maintenance needs and improving the efficiency of troubleshooting issues.

Mitigating biases might aid in glitch rejection. In Sec. III A 3, we will obtain all the triggers generated by Sage and PyCBC during glitch events in one month of O3a noise and prove that Sage is capable of effectively rejecting glitches. We also argue about the potential benefits of mitigating biases for glitch rejection.

OOD PSDs are good for classification. In Sec. III A 4, we will show that using a broader range of noise PSDs for the training data, which could even be OOD to the testing dataset, greatly aids in detection performance.

Glitches make classification easier. We show in Sec. III A 4 that models trained with real detector noise produce more confident classifiers than models trained with simulated colored Gaussian noise.

Parameter efficiency via inductive biases. Via an ablation study (see Sec. III C), we show that a model that is larger than Sage in terms of number of trainable parameters is not able to achieve nearly the same detection sensitivity as Sage, *if* it does not have the proper inductive biases.

II. METHODS

In this section, we elaborate on our Sage pipeline, which enables the detection of GWs at an efficiency that rivals the production-level search algorithm PyCBC in real detector noise. We provide effective mitigation tactics and training strategies that aid in mitigating the biases mentioned in Sec. I C concurrently. Mechanisms that are requisite for full reproducibility are explained in detail. All code and implementations are available [75].

Figure 2 shows a flowchart of the entire methodology for Sage. The following sections have been arranged in the order of the decision-making process. The entire methodology is fine-tuned based on prior knowledge about the testing dataset.

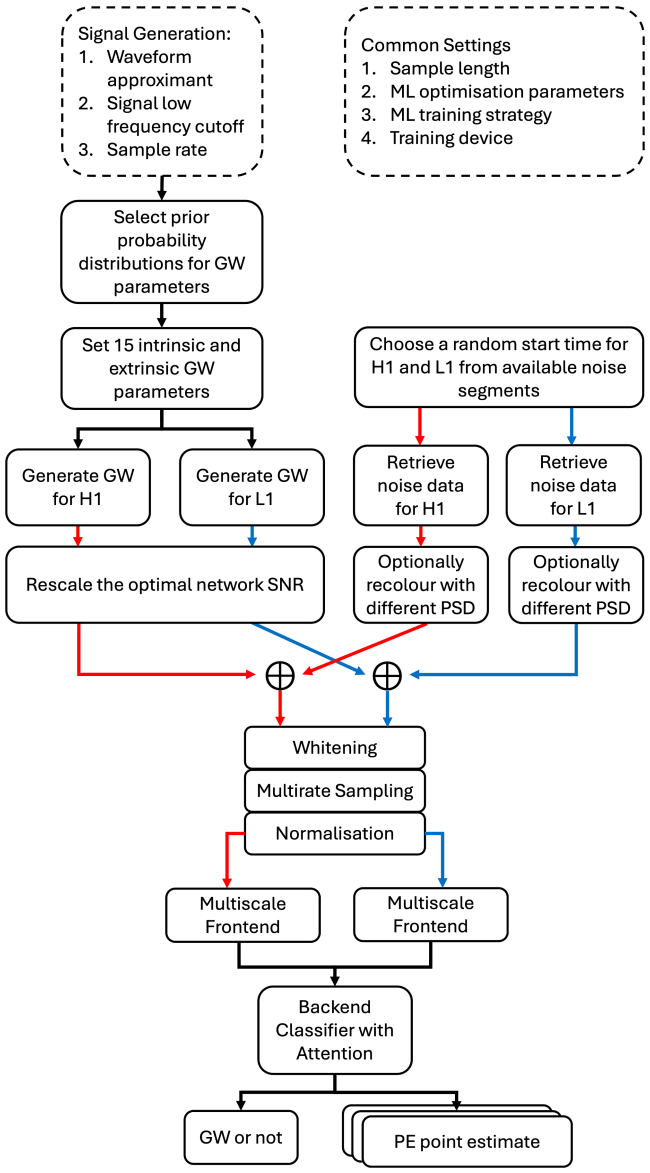


FIG. 2. Flowchart of Sage methodology. The details in the dotted box are set once per run; the signal generation box is specific for generating the signal class; the common settings apply for both the signal and noise class. All boxes with solid lines are iterated over during the training process.

TABLE I. Summary of testing dataset GW parameter distributions used for intrinsic and extrinsic GW parameters. This corresponds to the D4 realistic dataset of the MLGWSC-1 [21].

Notation	Parameter	Parameter distribution	Constraint
m_1, m_2	Masses	$m_1, m_2 \in U(7.0, 50.0)M_\odot$	$m_1 > m_2$
χ	Spins	$\chi \in U(0.0, 0.99)$	Distributed isotropically
ι	Inclination	$\cos\iota \in U(-1, 1)$	...
Φ_0	Coalescence phase	$\Phi_0 \in U(0, 2\pi)$	...
Ψ	Polarization	$\Psi_0 \in U(0, 2\pi)$	...
α	Right ascension	$\alpha \in U(0, 2\pi)$	...
δ	Declination	$\sin\delta \in U(-1, 1)$	...
d_c	Chirp distance	$d_c^2 \in U(130^2, 350^2)\mathrm{Mpc}^2$	...

A. Testing dataset parameter distributions

Any and all prior knowledge about the testing data is crucial in determining the GW parameter distributions for the training dataset, the network architecture, and the training strategies required for efficient learning.

Testing dataset configuration. A summary of the testing data parameter distributions used for all GW parameters is provided in Table I. The signals are generated as described in the realistic "dataset 4" (D4) of the MLGWSC-1 [21]. We use the IMRPhenomXPHM waveform approximant [76] that includes the effect of precession and higher-order modes. All higher-order (l, m) modes available in IMRPhenomXPHM are included, namely, (2, 2), $(2, -2)$, (2, 1), $(2, -1)$, (3, 3), $(3, -3)$, (3, 2), $(3, -2)$, (4, 4), and $(4, -4)$. Signals have a low-frequency cutoff of 20 Hz, and at least 33% of them should have an optimal network SNR below the threshold of 4 [21]. Figure 21 in Appendix A shows the 1D histograms of all relevant intrinsic and extrinsic parameters of 10^6 signals, using Table I for the parameter distributions. A *sample* in the testing dataset is defined as the time series data from a two-detector network (H1 and L1).

PyCBC only uses the fundamental mode and does not consider spin-induced precession effects in MLGWSC-1 [21]. If we were to include these effects in the training dataset configuration, gains in detection efficiency compared to PyCBC seen in our evaluation of the testing dataset could be attributed to this disparity. However, this is *contingent* on the detectability of the two effects in the testing dataset.

Detectability of higher-order modes. The relative power of higher-order modes is primarily dependent on, and proportional to, mass ratio ($q = m_1/m_2$) and total mass (M). Ignoring spins and the dependency on orientation, the detectability of higher-order modes relative to the $l = 2$, $|m| = 2$ fundamental mode for an $\mathrm{SNR} = 20$ signal is provided in [77]. All measures were estimated for a single LIGO detector with a sensitivity comparable to that achieved during the third observing run. The parameter space in Table I dictates that generating a signal with an optimal network SNR of 10 is in the ≈99th percentile and a mass ratio of 2 is in the ≈65th. Given the statistical improbability of generating a high-SNR, high mass ratio signal with sufficient total mass, we can expect negligible influence due to higher-order modes on the detection efficiency of our model.

Detectability of precession. The impact of varying different GW parameters on the detectability of precession is presented in [78]. They conduct their analyses assuming O2 detector sensitivities for LIGO but show that their results are unlikely to be affected when considering O3 sensitivities. Their results indicate that the recovered posterior distribution of precessing spin for a low SNR ($= 10$) signal resembles the prior, indicating the non-informativeness of the noisy data. Moreover, a signal SNR of 20–30 is shown to be required for a fifth percentile of the precession SNR distribution to be marginally detectable. This shows that the testing dataset is unlikely to have any signals with detectable precession effects.

The noise characteristics, although not analytic in the same sense as signals, can be summarized using the behavior of the PSDs and the non-Gaussian transient noise artifacts. MLGWSC-1 provides approximately 81 days of coincident real noise for the LIGO Hanford (H1) and LIGO Livingston (L1) detectors from the O3a observing run [79]. This consists of several noise segments that coincide in GPS time between H1 and L1 (referred to as time-correlated hereafter), with a duration of at least 2 h. All segments are 10 s away from any detection listed in GWTC-2 [79]. They use coincident noise segments that span a total of 30 days (day 1 to day 30 inclusive), between GPS time 1238205077 (2nd of April 2019, 01:50:59) to 1244605921 (15th of June 2019, 03:51:43) to create the "background" testing dataset. Signals were injected every 24–30 s into the background to form the "foreground" testing dataset. The parameters of the injected signals and their times of coalescence within the testing dataset noise are fixed using the seed provided by MLGWSC-1 for D4. All noise segments were ensured to have the data flag[2] and to not have any of the CBC_CAT1, CBC_CAT2,

[2]Definition of the flag provided [80].

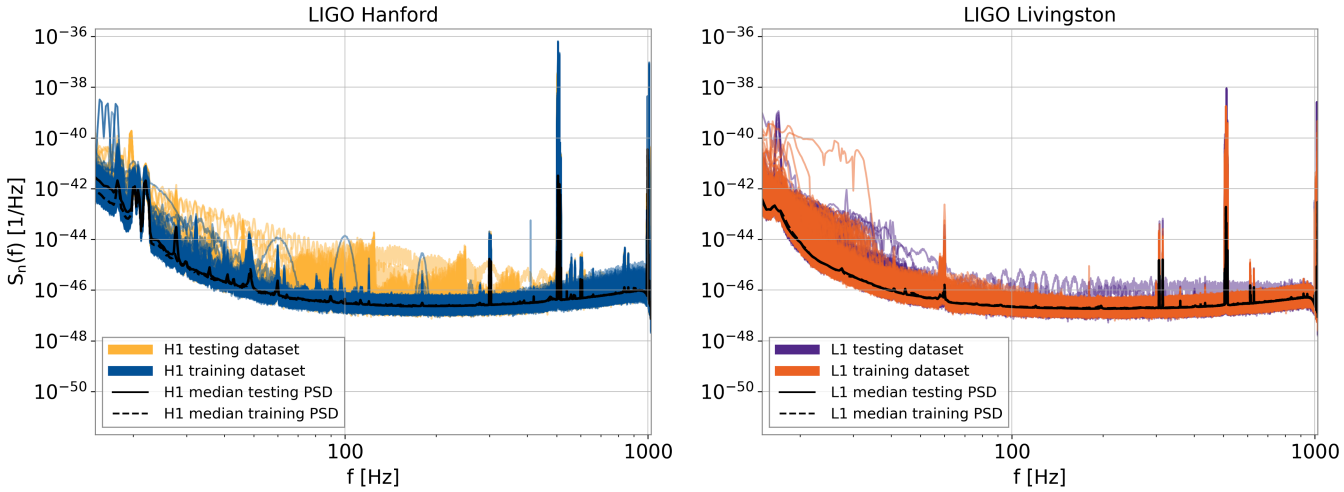


FIG. 3. Amplitude of estimated strain power as a function of frequency for different segments of O3a noise. We compare the estimated PSDs in the training (blue and orange curves) and testing (yellow and purple) datasets for the LIGO Hanford and Livingston detectors. Both plots have an f_{low} of 15 Hz at the noise low-frequency cutoff and an f_{high} of 1024 Hz at the Nyquist limit of the sampling rate. The solid and dashed black lines show the median PSD of all estimated PSDs in the testing and training dataset, respectively.

CBC_HW_INJ, or BURST_HW_INJ flags. The noise has a low-frequency cutoff of 15 Hz and is downsampled to a sampling rate of 2048 Hz. As a result of choosing noise segments over a long period of time, the PSD should vary by a non-negligible amount in the testing dataset. Figure 3 shows estimated PSDs of every 15 s of the 30 days of the background testing data using the Welch method [81], alongside the median of all estimated PSDs. Compared to the median, we see some outlier PSD realizations that are likely due to transient non-Gaussian noise artifacts but could also be caused by other known or unknown factors affecting the detector behavior.

The frequency of non-Gaussian noise artifacts was obtained by comparing the glitch event times reported in Gravity Spy [82] to the GPS times of the noise segments used for testing. A total of 16 531 and 28 739 glitches were present in the H1 and L1 O3a testing data, respectively. We can validate the glitch rejection capabilities of Sage compared to PyCBC using this information. Any conclusions obtained from our analysis are strictly under the caveat that the evaluation was performed on a limited testing dataset. We leave an in-depth investigation into the glitch rejection and PSD handling capabilities of Sage to the future.

We primarily evaluate Sage on the realistic D4 dataset described above. However, to experiment on a testing dataset without glitches and significantly varying noise PSDs, we also use the "dataset 3" (D3) testing dataset from the MLGWSC-1. All aspects of the methodology are tailored to D4, and any changes made to accommodate D3 will be explicitly mentioned in the relevant sections.

Modifications for D3. This dataset uses the same GW parameter distributions for generating the signals but uses 30 days of simulated colored Gaussian noise instead of real O3a noise. The 30 days of noise is split into several segments that are at least 2 h in length. For a given detector, each noise segment is colored using one of 20 noise PSDs estimated from real O3a noise for that detector.[3] We use the seed provided in the challenge paper to produce the testing dataset for a fair comparison to the D3 results submitted by PyCBC and ensure that we use a different seed for training Sage.

B. Training dataset parameter distributions

It is typical and necessary in supervised machine-learning to choose a feature distribution during training that closely matches the unseen testing features, for optimal generalization. For the most part, we abide by the waveform generation parameters and testing dataset parameter distributions mentioned in Sec. II A. We will use this section to provide justification for any deviations. Although we argued in Sec. II A that neither precession nor higher-order modes are detectable in the testing dataset, it does not warrant the usage of a waveform model like IMRPhenomD, which does not account for those effects, for training. Although the usage of a more accurate waveform model, like IMRPhenomPv2 or IMRPhenomXPHM, will take some unnecessary learning capacity from the model, when evaluated on the proposed testing dataset, the additional GW domain knowledge allows the model to more effectively reject samples from the realistic noise class. In the interest of preserving computational resources, we use IMRPhenomPv2, which is less expensive, instead of IMRPhenomXPHM. In order to facilitate this usage, it is necessary to employ an SNR distribution that operates

[3]PSDs provided in the MLGWSC-1 GitHub repository [83].

in the regime of generating signals where precession is detectable. We use this necessity as a gateway argument to further investigate the requirements of the training dataset parameter distributions.

As mentioned in Secs. I C 2 and II A, using the testing dataset parameter distributions for training is not a sufficient condition to guarantee that the model learns an unbiased representation of the required parameter space. For example, the sample distribution of a given parameter and the relative sample difficulty within the parameter bounds will heavily affect the generalization outcome. However, given the number of intrinsic and extrinsic GW parameters, it is not trivial to determine the joint distribution that ensures the fastest possible convergence of a training model to an unbiased representation.

As an empirical alternative, we study the model's testing performance by training on the joint distribution of the testing dataset described in MLGWSC-1 (reproduced in Table I) and alter the probability distribution function of the parameters over which the model exhibits learning bias. We define the joint parameter distribution of GW parameters that leads to the fastest convergence of a given ML model to its global minimum, without the biases due to limited sample and limited feature representation, as the optimal joint parameter distribution. Through experimentation, we determined chirp mass, mass ratio, and SNR as being the central source of such biases and altered their sample distributions in multiple intuitive ways to try to move closer to the optimal joint parameter distribution. The upper and lower bounds for the chirp mass and mass ratio distributions remain unchanged between the training and testing datasets, but we do change the bounds on network optimal SNR for the training distribution, which is reflected on the distance parameter.

Chirp mass $\mathcal{M}_c$ and mass ratio q are two independent mass coordinates, which we can analytically invert to obtain the component masses. This allows us to choose any arbitrary probability density $p(\mathcal{M}_c)$ and $p(q)$ and populate the $(\mathcal{M}_c, q)$ space within the required bounds (analytically derived from the bounds on m_1 and m_2). We can then analytically invert from $(\mathcal{M}_c, q)$ to (m_1, m_2) and apply rejection sampling based on the constraints: $m_1 > m_2$, $m_{\min} < m_1 < m_{\max}$, and $m_{\min} < m_2 < m_{\max}$. This procedure can be exploited to experiment with any chosen mass coordinate pair. For experimentation, we chose various realizations of the power-law and beta distributions alongside the uniform distribution for some chosen mass coordinate pair. To address the arguments in Sec. I C 2 about a potential bias against signals with longer signal duration, we experimented with the (τ_0, q) and (τ_0, τ_3) pairs as well. Here, we redefine τ_0 from Eq. (4) in terms of M and η and define τ_3 as

$$\tau_0 = \frac{5}{256\pi f_L \eta}(\pi M f_L)^{-5/3}, \tag{6}$$

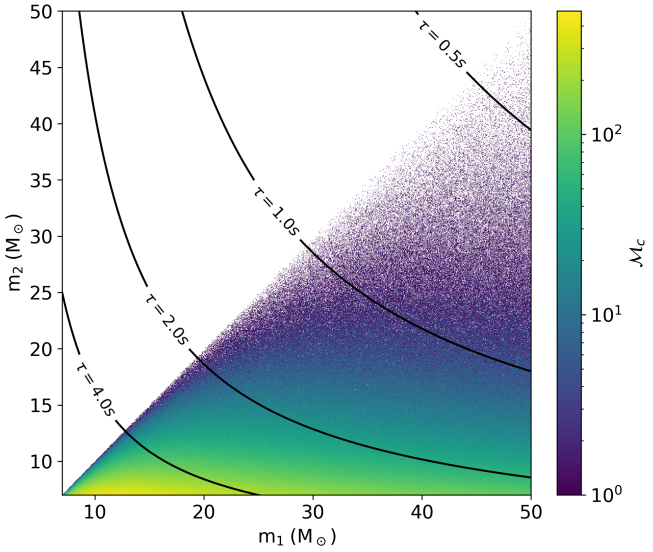


FIG. 4. 2D histogram of the joint distribution of component masses, $\mathrm{P}(m_1, m_2)$ used to generate the training dataset. 10^8 points were uniformly sampled from $U(\tau_0, \tau_3)$; points that lie within the mass bounds were accepted (≈13.4%) and the rest were rejected.

$$\tau_3 = \frac{1}{8 f_L \eta}(\pi M f_L)^{-2/3}. \tag{7}$$

We chose the (τ_0, τ_3) pair to investigate the use of the template density distribution inspired by the metric based on the SPA model [84]. The choice of chirp times τ_0 and τ_3 as coordinates on the parameter space comes with the benefit of the metric being constant at 1PN and essentially near-constant for higher PN orders in the local vicinity of every point in the parameter space and almost constant throughout [85,86]. Moreover, using the (τ_0, τ_3) pair comes with the added benefit of simple analytical inversion with the (M, η) pair [86]. If $p(\tau_0, \tau_3)$ were uniform within the bounds imposed by Table I, it would translate to an (m_1, m_2) space as shown in Fig. 4.

Finally, the network optimal SNR distribution is varied to conform to a uniform distribution, a half-normal distribution, or different variations of a beta distribution. The lower bound on SNR is set to 5, and the upper bound is allowed to differ with experiments but is always ≥15. Change to the SNR is assumed to reflect only on the distance parameter. Using a lower bound of 5 ensures a lower overlap with the noise class and a significant reduction in noisy labels. The upper bound must be high enough for the network to learn the scaling transformation of a signal resulting from a change in SNR. However, we also found that the network does not require as many high-SNR signals in the training dataset as low-SNR. A uniform distribution for SNR, for example, biased the network against learning low-SNR samples. For our experiments, we typically choose a parameter distribution that is skewed toward low SNRs.

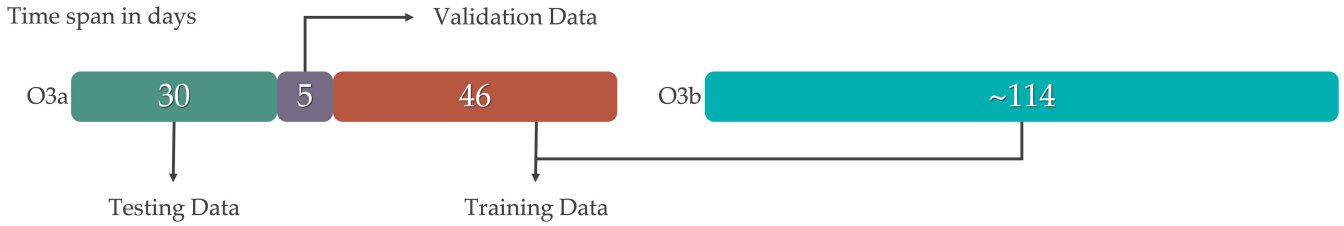


FIG. 5. Time spans (in days) for the training, validation, and testing datasets taken from the O3a and O3b observing runs for the H1 and L1 detectors. The O3a noise is provided by the MLGWSC-1 [21] and is time correlated between the H1 and L1. We downloaded the O3b noise from GWOSC, and it is not time correlated.

It is essential to mention that any significant results obtained from the above experimentation is only the best result from a limited set of joint distribution realizations, and it may or may not be close to the optimal joint parameter distribution.

The testing dataset parameter distributions dictate that the longest possible waveform that could be present in the unseen data will be ≈12.74 s in length for a low-frequency cutoff of 20 Hz, ignoring the effects due to higher-order modes and precession. This was calculated using SimIMRPhenomDChirpTime in LALSIMULATION with a 10% margin of error. Our primary results in Sec. III all use a 12-s sample length for the training dataset, which should contain the majority of the signal SNR for all considered signals. Since the signal duration could be longer when considering more realistic estimates of chirp time, we conducted an experiment with a 20-s sample length as well. Using a 20-s sample length would encompass 100% of the signal SNR but would require longer training and evaluation times while also increasing the problem difficulty. We found that using a 12-s length was sufficient to learn the parameter space well.

With regard to noise, we have the latter 51 days (day 31 to day 81 inclusive) of the provided 81 days of O3a noise via MLGWSC-1 that is not used for the testing dataset, GPS time 1244608795 (15th of June 2019, 04:39:37) to GPS time 1253966055 (1st of October 2019, 11:53:57). We use the first 5 days of the 51 days as the validation dataset. The rest of the data (day 36 to day 81 inclusive), alongside ≈114 days and ≈113 days of H1 and L1 O3b noise, is used for training the model (refer to Fig. 5). The downloaded O3b noise segments span between GPS time 1256655618 (1st of November 2019, 15:00:00) to GPS time 1269363618 (27th of March 2020, 17:00:00). They have a duration of at least 1 h and are 30 s away from any detection listed in GWOSC.[4] All noise segments were ensured to have the data flag and to not have any of the CBC_CAT1, CBC_CAT2, CBC_HW_INJ, or BURST_HW_INJ flags. The noise segments provided by the MLGWSC-1 for O3a noise were constrained to coincide in GPS time between the two detectors, which limits the total amount of data used for training. Given that time correlation is not a requirement for training, the O3b noise segments that we downloaded are not constrained in this manner. More information on how this data is used to create the training samples is provided in Sec. II C.

Similar to the testing dataset, we estimated the PSDs for the entirety of the training data, and the results are compared in Fig. 3 for both detectors. It was a cause for concern to notice several outlier PSDs in the H1 testing dataset that had higher energies in the relevant frequency range when compared to the training dataset. These PSDs in the testing dataset can be considered OOD, and the model might struggle during testing when encountering them. There is ample precedence for this expectation in the machine-learning literature [89,90]. Since the noise PSDs and glitches are different between the training and testing datasets, validating the model on a fixed set of noise realizations will bias our choice of the best-performing network instance. A significant overlap between the training and validation set will hinder the ability to detect the onset of overfitting, and using the testing dataset to validate will again bias our choice of the best-performing network. The five-day validation dataset is not the best choice given these biases but will still be useful in detecting signs of overfitting.

Modifications for D3. To evaluate Sage on the D3 testing dataset, we train a separate ML model using simulated colored Gaussian noise instead of real O3a noise.

Biases addressed. We address the *bias due to limited sample or feature representation* by choosing an appropriate training dataset parameter distribution for chirp mass, mass ratio, and SNR. For example, signals with longer duration or higher mass ratio are represented much better in $U(\tau_0, \tau_3)$ than $U(m_1, m_2)$ mass distribution. *Bias due to sample difficulty* can be addressed by choosing the optimal SNR distribution to be a half-normal distribution, which places a higher weight on the harder-to-classify low-SNR signals. Placing the lower bound for optimal SNR at 5 reduces the *bias due to class overlap*. Using a 12-s sample length addresses the *bias due to insufficient information*, and using a broad enough parameter space for SNR addresses *bias due to lack of variation*. Finally, using a larger amount of noise leads to a broader training

[4]Event list obtained from [87] and noise segment metadata obtained from [88].

distribution for noise PSDs and glitches and addresses the *bias due to lack of variation*.

C. Data generation

Generation of signal samples. All previous studies on machine-learning-based GW detection have used a fixed dataset size to train their model. Our investigations on mitigating learning bias showed that IID training dataset size plays an equal role alongside the joint sample distribution of the GW parameters. We use PyCBC to generate frequency domain IMRPhenomPv2 waveforms on the fly during the training phase of our model. This allows for unlimited signal data, constrained only by the number of training epochs, and is equivalent to augmentation on all GW parameters. Each sample is injected into a noise realization, the generation of which is described in the remainder of this section.

Generation of noise samples. Unlike signals, we do not have the liberty to produce unlimited IID samples from ≈160 days of coincident detector noise. However, we can obtain unlimited non-IID noise realizations as a near-ideal proxy. Inspired by the time-slides method for background estimation in a network of GW detectors [91], we select start times independently for each detector from a uniform distribution over the entire ≈160-day span of available data. These samples may or may not be time correlated and result in a large number of non-IID noise realizations. However, due to the limited number of non-Gaussian noise transients and varying PSD realizations, the advantage that the model gains from this method will eventually saturate. An estimate of the number of IID samples will provide us with some insight on when this will happen. At a sampling frequency of 2048 Hz, 160 days of noise gives us access to $\approx 1.15 \times 10^6$ IID samples. Between two detectors, there are $\approx 1.33 \times 10^{12}$ noise realizations, which is the Cartesian product of two sets of $\approx 1.15 \times 10^6$ samples in H1 and L1 (essentially time shifting the detector data by more than the sample length of 12 s). The final number depends on noise augmentation and is provided in Sec. II D.

Modifications for D3. Signal generation is exactly the same for D3 and D4. For noise generation, we first produce white Gaussian noise with a different seed for each sample. Each noise sample of the noise class and additive noise in the signal class is colored using one of the 20 estimated noise PSDs that is used for creating the D3 testing dataset. This nullifies the difference between the training and testing noise PSD distribution, meaning the trained model will not encounter any OOD noise samples during testing.

Biases addressed. Using on-the-fly signal generation is equivalent to augmenting on all GW parameters and addresses the *bias due to lack of variation* and *bias due to limited feature representation*. This increases the number of discrete points within the continuous transformation space of all GW parameters and allows the network to learn the transformation well. This bias is also addressed for noise by using random start times for each detector.

D. Data augmentation

Signal augmentation. Since we use on-the-fly signal generation, which essentially augments on all GW parameters, it is not necessary to include any explicit form of signal augmentation. The optimal network SNR is, however, shifted and rescaled in accordance with probability distributions mentioned in Sec. II B. This action effectively alters the distance and, thereby, the chirp distance distribution. Any correlation that the original optimal network SNR had with the other GW parameters is lost. Although there is some loss of astrophysical prior knowledge, the null correlation should help to simplify the data manifold of the signal class. This reduces the *bias due to sample difficulty* affecting the learning of low-SNR regions of the parameter space. For example, binary systems with edge-on inclinations will typically have lower SNRs than face-on systems. Without removing this correlation, the network will be biased against learning edge-on systems. To show the benefit of on-the-fly signal generation, Sage also includes limited signal augmentation procedures for the sake of comparison—after generating a fixed size training dataset of h_+ and $h_\times$ signal components, we randomly choose the polarization and sky position based on the distributions provided in Table I, before projecting the components into $h(t)$.

Noise augmentation. The noise sample generation procedure explained in Sec. II C innately includes noise augmentation via the choice of random start times and the non-necessity of time correlation between detectors. Given that not all IID noise samples are significantly different, the potency of this augmentation to lower the training loss eventually saturates. However, we postpone this saturation effect by optionally recoloring (with a 40% probability of being recolored) the noise realization of each detector with a precomputed set of noise PSDs from the ≈160 days of O3 training noise. We compute a total of 250 000 IID PSDs using 15-s nonoverlapping noise segments for each detector using the Welch method.

To investigate any changes in detection efficiency during the testing phase due to the lack of training on OOD PSDs seen in Fig. 3, we also precompute a set of 172 800 PSDs from the 30 days of O3a testing data (again using 15-s nonoverlapping noise segments). Any increase in detection efficiency observed during testing due to augmentation using these PSDs during training could indicate the lack of proper generalization to OOD PSDs. During the recoloring stage, we also have the option to shift the entire PSD up or down to either increase or decrease the overall noise power.

Previously, in Sec. II C, we showed that our noise generation procedure has access to $\approx 1.33 \times 10^{12}$ IID noise realizations. Given that the recoloring procedure with ≈160 days of O3 training noise has 250 000 IID PSD

realizations in each detector, this number increases to $\approx 8.26 \times 10^{22}$. With 172 800 IID PSDs, this number will be $\approx 3.95 \times 10^{22}$. The noise samples will be exactly the same as seen in O3 when they are time-correlated between H1 and L1, and no recoloring is applied. Under this condition, we must include the number of non-IID noise realizations to calculate the probability that the network sees such samples (following the same procedure as for IID samples, the number of non-IID samples is $\approx 6 \times 10^{31}$). The probability of seeing an exact O3 noise realization during training is, thus, $\approx 1.4 \times 10^{-22}$ for recolored noise samples and $\approx 8.6 \times 10^{-11}$ for nonrecolored samples.

Modifications for D3. Signal augmentation is consistent between the models trained for D3 and D4. None of the above noise augmentation procedures apply to D3.

Biases addressed. Using the recoloring augmentation further increases the amount of variation in the training noise features and addresses the *bias due to lack of variation* by broadening the training noise feature distribution. Using an estimate of the testing dataset PSDs to recolor the training dataset introduces a mild train-test overlap but addresses the *bias due to train-test mismatch*.

E. Data transformation

Whitening. Once the class sample (either signal or noise) is generated and augmented, we whiten it using a single, precomputed PSD for each detector, which remains constant across all samples. This PSD is estimated by taking the median over several noise PSDs from O3 training data and smoothed via inverse spectrum truncation using a Hanning window. To whiten the data, we divide the sample in the Fourier domain by the smoothed noise PSD, which standardizes the amplitude distribution across frequencies, ensuring that the transformed data have approximately equal variance across the spectrum. Using a fixed PSD for whitening all samples instead of estimating the PSD for every sample results in mildly nonuniform power between different frequencies. We observed that the presence of this nonuniformity in the training data did not contribute to any noticeable degradation of the network when evaluated on the testing data, thus voiding the need to estimate the PSD for every input sample. For D3, we compute the median PSD from the 20 provided PSDs for each detector, which is then used to whiten every input sample.

Multirate sampling. Each input, being 12 s in duration at a sampling frequency of $f_s = 2048$ Hz, has 24576 points. Training a neural network on such a long sample has two main disadvantages: (i) It demands a substantial computational cost, especially for complex problems requiring large datasets, and (ii) most standard architectures will struggle to formulate long-range dependencies due to an insufficient effective maximum receptive field [72]. It is, therefore, crucial to compress the data to its permissible limits. The solid red line in Fig. 6 indicates the Nyquist limit of the input data, $0.5 f_s$, alongside the frequency evolution curves

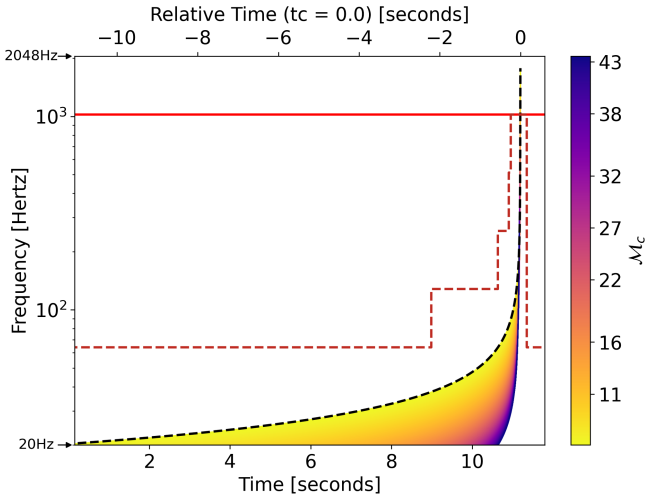


FIG. 6. Frequency vs time curves of GWs generated from the testing dataset parameter distributions described in Table I colored by chirp mass. The dashed black line shows the frequency profile of a GW with the lowest possible chirp mass or longest duration. The solid red line shows the Nyquist limit of the generated data at 1024 Hz given that the sampling rate is 2048 Hz. The dashed red line shows the modified Nyquist limit after applying the multirate sampling algorithm.

of the chirp mass distribution for the training data. Given that the time of coalescence (t_c) of any signal is placed within a predefined time range, it is straightforward to see the inefficiency in using an unnecessarily high f_s to represent regions in time where we are *a priori* aware of the maximum possible GW frequency.

We take inspiration from [92] that implements multirate sampling for the detection of binary neutron star mergers. The goal is to modify the Nyquist limit at different points along the given time series according to the maximum possible instantaneous GW frequency dictated by the training GW parameter distributions. This can be seen as a form of physics-informed time series compression. Consistency for all training and testing samples can be guaranteed with the following bottleneck criterion.

Bottleneck 1. For a 12-s sample length, we choose the training distribution for t_c to be $U[11.0, 11.2]$ s. Let us assume that the frequency evolution profile of the signals in the training distribution is well described by Eq. (4). At any given time instant till the merger, the instantaneous frequency of the longest possible signal placed at $t_c = 11.0$ s will be higher than that of any other signal in the training distribution. Thus, if we modify the Nyquist limit to lie just above the frequency profile of this signal, it will ensure that *all* possible GW signals in the training distribution are captured completely. In the time domain, we can split the signal into time bands of constant sampling frequency above the minimum allowed value.

Say we place the longest possible signal in the training distribution, described by the lowest possible chirp mass (to first order), at $t_c = 11.0$ s. We move back in time along

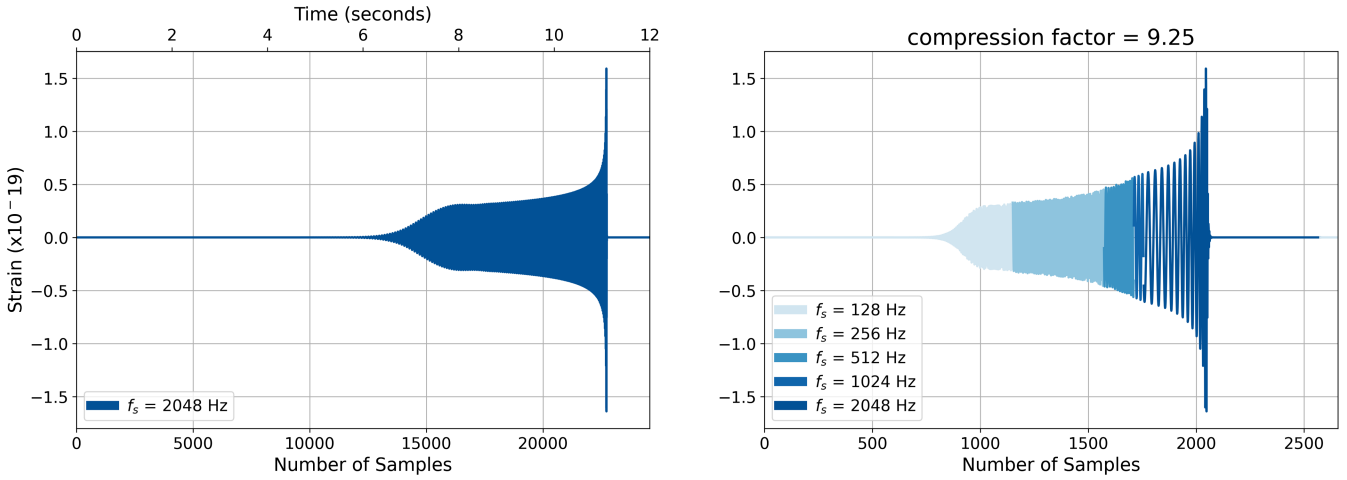


FIG. 7. Example application of multirate sampling on an IMRPhenomXPHM BBH template with the following parameters: $m_1 = 20M_\odot$, $m_2 = 10M_\odot$, zero spins, and a time of coalescence of 11.1 s. The colors depict the sampling frequency (f_s) of different sections of the compressed signal. $f_s = 2048$ Hz corresponds to an unaltered section of the generated waveform.

the frequency profile of this signal, starting from $f^0_{\rm gw} = f_{\rm ISCO}$, where $f_{\rm ISCO}$ is the instantaneous frequency at the innermost stable circular orbit (ISCO) of a single black hole whose mass is a sum of the source components and $f^i_{\rm gw}$ is the maximum GW frequency in time band i. We use $f_{\rm ISCO}$ as a proxy for the frequency at merger. We start at a sampling frequency $f^0_s = 1024$ Hz at $f_{\rm ISCO} = 314.1$ Hz, where f^i_s is the constant sampling frequency within time band i. The sampling frequency is halved $f^i_s = f^{i-1}_s/2$ for every corresponding halving of the GW instantaneous frequency $(f^{i-1}_{\rm gw}/f^i_{\rm gw}) - 2 \approx 0$. This process is repeated until we reach the lowest allowed sampling frequency at 128 Hz. The region beyond $f_{\rm ISCO}$ can contain GW frequency components that are high enough to require an f_s of 2048 Hz. Thus, the time instant at $f^0_{\rm gw}$ defines the lower boundary of the $f_s = 2048$ Hz time band.

Bottleneck 2. The second bottleneck is defined by the ringdown of the GW signal, with the highest total mass $M = m_{\rm max} + m_{\rm max} = 100M_\odot$ with the maximum allowed spin placed at the upper bound of t_c at 11.2 s. This configuration gives us the longest possible ringdown signal dictated by the quasinormal mode damping time at a given spin and mode (we use only $l = 2$ and $m = 2$) and typically consists of high-frequency components. Thus, we set the highest available f_s of 2048 Hz for this region. This defines the upper boundary of the $f_s = 2048$ Hz time band.

Bottleneck 3. Additionally, we impose an offset to each boundary due to t_c being measured in geocentric time. This accounts for the t_c in detector time being outside the two boundaries. The final boundaries for the $f_s = 2048$ Hz time band are $\approx[10.95, 11.35]$ s.

The region beyond 11.35 s is assumed to be pure noise and is sampled at 128 Hz. Figure 6 shows the frequency profile of the longest possible GW as a dashed black line. The modified Nyquist limit is shown as a dashed red line. Some high-frequency features close to t_c will not be registered due to the maximum possible f_s, as seen in Fig. 6, but the loss in SNR should be negligible. Figure 7 shows an example of the procedure applied to a GW within the considered parameter space.

Normalization. Each multirate sampled time series is treated independently and normalized to lie within the range of $[-1, +1]$. We also use an input standardization layer within the network architecture that rescales the features to ensure a zero mean and unit standard deviation. Say we have a batch with dimensions (N, C, L), where N is the batch size, C is the number of channels or detectors, and L is the length of the input sequence. BatchNorm uses all samples in (N, L) and standardizes each detector data C_i separately. This would work well in simulated Gaussian noise, but the presence of an outlier, like a loud glitch, in one of the batch elements for a particular detector will heavily influence the normalization for that detector in all batch elements. LayerNorm uses all samples in (C, L) and standardizes each batch element N_i separately. This is equivalent to standardizing the time-correlated strains from the two detectors together. LayerNorm will bias the network against detectors that typically have lower strain amplitudes. InstanceNorm standardizes each individual strain data from each detector independently. This is the most suited layer for our application, and we use it to transform each time series before providing it to the main model architecture.

Biases addressed. Whitening reduces the *bias due to limited feature representation* and *spectral bias* by normalizing the power due to different frequencies. Multirate sampling reduces the sample length and negates the need for a larger receptive field to learn low chirp mass signals (aids in learning long-term dependencies). This addresses the *bias due to insufficient information* and *bias due to sample difficulty.* Normalizing the amplitude to lie within

$[-1, +1]$ also reduces to *bias due to limited feature representation*, similar to whitening.

F. Optimization

Multiobjective loss function. We use the standard binary cross entropy (BCE) loss for the purpose of classifying a sample as either a noisy signal or pure noise. The unreduced form of the BCE loss for a batch of network outputs $\mathbf{x}$ and targets $\mathbf{y}$ can be written as

$$\mathcal{L}^{\mathrm{gw}}_{bce}(\mathbf{x}, \mathbf{y}) = \{l_1, \ldots, l_n\}^\top,$$
$$l_i = -w_i[y_i \cdot \log x_i + (1 - y_i) \cdot \log(1 - x_i)],$$
$$\text{where } i = 1, \ldots, n,$$

and, for each ith batch element (for a batch size of n), x_i and y_i are the network output and target, l_i is the loss, and w_i is a manual rescaling weight. The unreduced loss $\mathcal{L}^{\mathrm{gw}}_{bce}(\mathbf{x}, \mathbf{y})$ is reduced as $\overline{\mathcal{L}^{\mathrm{gw}}_{bce}(\mathbf{x}, \mathbf{y})}$ for every batch (the bar represents sample mean). There is a possibility for one of the log terms to be mathematically undefined or for the network to tend to an infinite gradient. PyTorch deals with this possibility by clamping its log function outputs to be greater than or equal to -100. Thus, the loss function is always finite, and backpropagation is linear. Alongside the BCE loss, we also optimize on a mean-squared error (MSE) loss function for each GW parameter ϕ in t_c, $\mathcal{M}_c$. The unreduced form of this loss is given by

$$\mathcal{L}^{\phi}_{mse}(\mathbf{x}^\phi, \mathbf{y}^\phi) = \{l^\phi_1, \ldots, l^\phi_n\}^\top,$$
$$l^\phi_i = (x^\phi_i - y^\phi_i)^2,$$
$$\text{where } i = 1, \ldots, n,$$

and noise samples that do not have a target y^ϕ_i are ignored. For each ith batch element that is not a noise sample (for a batch size of n), x^ϕ_i and y^ϕ_i are the network output and normalized target, respectively, for the given GW parameter, and l^ϕ_i is the loss. Each network output x^ϕ_i is limited within the range [0, 1] using a Sigmoid function, and the unreduced loss $\mathcal{L}^{\phi}_{mse}(\mathbf{x}^\phi, \mathbf{y}^\phi)$ is reduced as $\overline{\mathcal{L}^{\phi}_{mse}(\mathbf{x}^\phi, \mathbf{y}^\phi)}$ for every batch. The primary purpose of these loss functions is *not* to obtain a point estimate of the parameters, but to instead regularize the classification problem by acting as a soft boundary condition. The final batch loss is given by

$$\mathcal{L} = \overline{\mathcal{L}^{\mathrm{gw}}_{bce}(\mathbf{x}, \mathbf{y})} + \sum_{\phi \in \{t_c, \mathcal{M}_c\}} \overline{\mathcal{L}^{\phi}_{mse}(\mathbf{x}^\phi, \mathbf{y}^\phi)}\,. \tag{8}$$

Optimization parameters. The learning dynamics is optimized using an Adam optimizer [93] with an initial learning rate of 2×10^{-4} and a weight decay of 10^{-6}. The learning rate is then scheduled using CosineAnnealingWarmRestarts that cycles every five epochs, with a minimum learning rate of 10^{-6}. We use gradient clipping set to a maximum allowed gradient norm of 1 and a batch size of 64 for all our runs.

Biases addressed. The MSE loss term $\overline{\mathcal{L}^{\phi}_{mse}(\mathbf{x}^\phi, \mathbf{y}^\phi)}$ provides an inductive bias to learn the GW signals properly. In other words, GW signals become easier to learn, thus addressing the *bias due to sample difficulty.*

G. Network architecture

Sage uses a bottom-up hierarchical architecture with two main stages. The first stage is a front-end multiscale feature extractor for each detector that highlights the essential regions and, more importantly, deemphasizes unwanted attributes at various scales in the time domain. The second stage is a back-end classifier powered by attention to form temporal relationships within each detector and to emphasize any correlation, or lack thereof, between detectors. There is ample precedence in the machine-learning literature to suggest that a multistage detection process that fragments the primary problem into multiple subproblems will lead to better generalization and faster convergence over single-stage approaches [94–96]. The rest of this section is dedicated to explaining parts of the two-stage architecture, their intended purposes, advantages, and necessity.

Front-end feature extractor. The front-end architecture is inspired by two main works[5]: (i) [98] that argues for the need for large kernels and provides design principles for learning contradictory tasks such as classification and localisation of features simultaneously and (ii) [99] that designs a multiscale residual block called the OSNet.

The key feature of the OSNet [99] is to simultaneously analyze an input over multiple scales to create a comprehensive representation. The scale, dictated by the effective receptive field, can be varied by stacking n_c convolutional layers, with a higher value of n_c resulting in larger receptive fields. Stacking small kernel convolutional layers was initially introduced as an efficient alternative to using a single large kernel, as it results in a similar effective receptive field with a lower computational cost. However, [98] argue and empirically prove that large kernels lead to better generalization in per-pixel classification tasks, compared to an equivalent stacking of small kernels. They conclude that the denser connection between large kernels and the output enables the network to handle input transformations more resiliently. Additionally, according to [98], to retain the ability to localize important features, the architecture must be fully convolutional and not contain any fully connected layers or global pooling layers.

[5]We also took inspiration from and had useful discussions with the G2Net 2021 Kaggle competition winners via private communications [97].

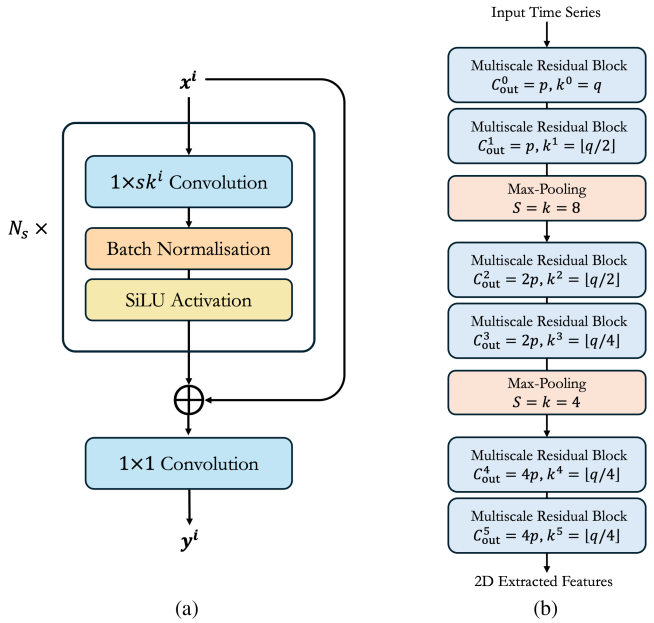


FIG. 8. A diagrammatic depiction of the multiscale front-end feature extractor. (a) shows a single multiscale residual block. The input x^i is analyzed separately and simultaneously by N_s different 1D large convolutional kernels with kernel size sk^i, where N_s is the number of scales and s is a set of constant prefactors applied to the base kernel size k^i. The $\bigoplus$ is an elementwise addition. (b) shows the full front-end architecture.

Based on these key insights, we build our front-end architecture. A diagram of our large kernel multiscale residual block is shown in Fig. 8(a). In Fig. 8(b), these blocks are then stacked with intermittent downsampling via 1D max pooling to create our full feature extractor. Although we downsample the features along the temporal dimension using a stride S, this is performed after generating C_{out} multiscale output channels, where S and C_{out} are chosen such that the 2D output feature map is larger overall than the input. Thus, we retain the dense connections between kernels and the extractor feature output.

Reproducibility. With reference to Fig. 8(a), the large 1D kernels used for the multiscale feature extractor uses $N_s = 5$ scales with prefactors $s = \{0.25, 0.5, 1, 2, 4\}$. In Fig. 8(b), the initial kernel size k^0 is set to $q = 64$, and the initial number of output channels C^0_{out} is set to $p = 32$. These values were fine-tuned such that they minimize the loss during the training phase.

The input time series for each detector is provided to a separate front-end feature extractor (Fig. 8) and there is no weight sharing at this point between detector data. This is done to ensure that subtle differences within the data collected from each detector can be learned without any interference. The small and long-range features extracted with the multiscale architecture should encode rich spectrotemporal information from the GWs, non-Gaussian noise artifacts, and varying noise PSDs.

Back-end classifier. We use a variation of the ResNet architecture [100] introduced in [101] which integrates the convolutional block attention module (CBAM) into the ResBlock of a ResNet. The ResNet typically comprises a stack of ResBlocks, with the output feature map of one block feeding into the next. The residual nature of each ResBlock refers to the architectural motif $y = \Phi_s(x) + x$, where Φ_s is some arbitrary neural network module acting on an input feature map x. The recasting of the input feature map x into $\Phi_s(x)$ is called a residual connection, and it allows for the model to mitigate the vanishing gradient problem [100].

The salient design feature of CBAM is to sequentially infer attention maps along the channel and spatial dimensions of $\Phi_s(x)$. Say we define the output of the arbitrary neural network module within the ResBlock $F = \Phi_s(x)$ as an intermediate feature map. CBAM applies the spatial and channelwise attention maps (M_s, M_c) to produce a final output $F'' = M_s(F') \otimes F'$, where $F' = M_c(F) \otimes F$ and $\otimes$ denotes elementwise multiplication. So y will be $F'' + x$ instead of $\Phi_s(x) + x$ (in reference to ResBlock motif). We run Sage on a CBAM-ResNet50 back-end variant for all experiments. The channelwise attention $M_c(F)$ is computed as

$$M_c(F) = \sigma[\Phi_c(P_{\text{avg}}(F)) + \Phi_c(P_{\text{max}}(F))], \tag{9}$$

where Φ is some multilayered perceptron (MLP), (P_{avg}, P_{max}) are average pooling and max pooling operations, respectively, and σ is the sigmoid activation function. The MLP has a single hidden layer that is smaller than the input, thus acting as a bottleneck, similar to an autoencoder architecture [102]. The spatial attention $M_s(F')$ is computed as

$$M_s(F') = \sigma[P_{\text{avg}}(F') \circledast P_{\text{max}}(F')], \tag{10}$$

where $\circledast$ is the convolution operation. The usage of pooling operations for the aggregation of information is well founded [103,104], and CBAM exploits this feature across the spatial and channel dimension and empirically proves that the usage of both pooling operations together greatly improves detection capability. The spatial attention will aid greatly in forming temporal relationships within the time series of each detector, and the channelwise attention will construct correlations between detectors or indicate a lack thereof.

Point-estimate outputs. The CBAM-ResNet50 is made to output a 512-dimensional embedding, which is passed through three separate (i.e., no shared weights) fully connected layers with 512 neurons each. Each layer is then responsible for one unnormalized output. A sigmoid activation function is then applied to each output to restrict it between [0, 1]. The three normalized outputs are point

estimates on the probability of a GW event, p_{GW}, and the normalized GW parameters $\{t_c, \mathcal{M}_c\}$. The normalized values are used to compute the loss function, and the unnormalized counterpart of p_{GW} is used as the output ranking statistic (also referred to as the network output). Through backpropagation, the 512-dimensional embedding should encode information about all outputs, given the composite loss function in Eq. (8). We choose chirp mass as it is the best-measured parameter for the most observed mergers, as it determines the leading-order GW radiation during the inspiral phase [105]. This acts as a soft constraint and provides an additional inductive bias to aid the network in distinguishing between the signal and noise classes. The time of coalescence estimate acts as a reference point and aids in feature localization. Furthermore, we can also use the t_c estimate as input to parameter estimation pipelines to investigate the generated triggers. An example of this estimate for parameters $\{t_c, \mathcal{M}_c\}$ is shown in Fig. 24 (see Appendix D) for two different Sage runs (see Sec. III A 1 for details about runs).

Biases addressed. The multiscale front end with large kernels is useful when learning time series signals with a high dynamic range of frequencies [98,106]. Spatial attention aids the model in learning amplitude, frequency, and phase evolution profiles by forming temporal relationships using the multiscale features. The attention mechanism automatically increases the weight of underrepresented scales, thus addressing *spectral bias*. The attention mechanism and multiscale architectures have been used before to address this bias, as seen in [107–109]. Channelwise attention aids in forming a correlation between the temporal relationships formed for the signal in each detector. The two attention mechanisms aid significantly in learning long-term dependencies [71,110,111], addressing the *bias due to sample difficulty*. Moreover, training a deeper network for longer (without overfitting) increases the effective receptive field, which also aids in learning long sequences [72].

III. RESULTS AND DISCUSSION

We will use this section to further describe the injection study, elaborate on the various training configurations employed, and compare our results with previous works on the same datasets (D3 and D4) [21,53,64]. Furthermore, we analyze the behavior of Sage under different conditions and discuss the validity of juxtaposing different results on established evaluation metrics. For the sake of completeness, we also investigate the performance of Sage by removing certain important components from our pipeline to understand its contribution to the overall system (also known as an ablation study).

We ran Sage on multiple training configurations to investigate the impact of different biases, time-varying PSDs, and glitches. Table II gives a brief overview of the runs that we will be discussing in this section. Any configuration that is not mentioned in this table is consistent between all runs and is as described in Sec. II. The steps taken to evaluate Sage after the end of training are provided here in sequential order.

Common settings. As mentioned in Secs. II B and II A, the training noise is O3a(day 36–51) and O3b(day 1–113), and the testing noise is O3a(day 1–30) for all our runs. All runs were trained with 80 million samples (1:1 ratio between the signal and noise class) split equally over 40 epochs.

Best trained model. The validation noise dataset will be slightly different, compared to the training and testing datasets, in terms of noise PSDs and glitches. Thus, we retrieve the network learnable parameters (i.e., weights and biases) that correspond to the five lowest epoch validation losses for testing. We then test the model on the "background" and "foreground" testing datasets discussed in Sec. II A for each set of network parameters and pick the one that detects the most signals at an FAR of one per month. This procedure is exactly what the Neyman-Pearson lemma prescribes, but to be clear we *do not* advocate for the number of detections to be a less-biased or better metric compared to any other.

TABLE II. Summary of all Sage runs discussed in the results section. Column 1 gives a shorthand for the discussed runs and will be used to describe a singular training instance of that type that achieved the highest sensitive distance at an FAR of one per month. Column 2 describes the distribution of component masses (m_1, m_2). The τ_0-τ_3 coordinate pair can be analytically inverted to the M-η pair, as described in Sec. II B, which can then be used to derive the distribution of component masses. The $\longrightarrow$ symbol refers to a particular training strategy of transitioning from one distribution to another (refer to the Sage—Annealed train run in Sec. III A 1 for more details). Finally, column 3 provides details about the noise segment used to obtain precomputed PSDs that are meant to recolor the training noise realizations.

Run name	Training data—Mass distribution	Noise segment for recoloring
Sage—Broad	$U(m_1, m_2)$	$\{\mathrm{O3a(day1-30)}\}$
Sage—Limited	$U(m_1, m_2)$	$\{\mathrm{O3a(day36-81)}, \mathrm{O3b(day1-113)}\}$
Sage—D4 metric	$U(\tau_0, \tau_3)$	$\{\mathrm{O3a(day1-30)}\}$
Sage—Annealed train	$U(m_1, m_2) \longrightarrow U(\tau_0, \tau_3)$	$\{\mathrm{O3a(day1-30)}\}$
Sage—D3 metric	$U(\tau_0, \tau_3)$	⋯

Testing iteration. The 30-day testing data is split as sequential windows with a step size of 0.1 s. Each window is whitened, multirate sampled, and normalized, consistent with the training dataset, prior to being provided as input to the trained network. The network outputs a ranking statistic for each 12-s window, which we assume originates due to a possible GW event with a time of coalescence within [11.0, 11.2] s (the bounds used during training). This is called a *trigger*. Given that the step size is 0.1 s, a GW event can have at most three triggers.

Trigger clustering. To identify independent gravitational-wave candidates, we cluster nearby triggers based on a predefined time threshold (2 s). We iterate through the list of triggers, grouping consecutive ones if their time difference is below the threshold; otherwise, a new cluster is initiated. Each cluster is then represented by its most significant trigger, determined by the maximum ranking statistic within the cluster. Any subsequent use of the term "trigger" must be interpreted as this independent cluster maxima. This approach ensures that multiple responses to the same underlying signal, or to closely spaced noise fluctuations, are not overcounted.

Labeling cluster maxima. Only triggers above a ranking statistic of 0 are retained (to reduce compute cost). No prior knowledge about the location of the injected events is taken into account when assigning the trigger a ranking statistic. For the foreground dataset, triggers within $[t_c^{\rm inj} - 0.2, t_c^{\rm inj} + 0.2]$ s of the 12-s window, where $t_c^{\rm inj}$ is the time of coalescence of any one of the injected signals, are labeled as *true positive* triggers for the corresponding injection (not to be confused with a binary ML label). For each true positive trigger, we get a point estimate of the time of coalescence and chirp mass from Sage (see Sec. II G). All other foreground triggers that do not correspond to a known event are labeled as *false positives*. For the background dataset, all triggers are labeled as false positives. The labels provided to the triggers are purely for the purpose of computing the following evaluation metrics.[6]

Evaluation metrics. The sensitive distance metric at different FARs is computed as described in Sec. I C 4. To compute receiver operating characteristic (ROC) curves, we use the true positive triggers for each signal from the foreground dataset and false positive triggers from arbitrary noise windows of the background dataset. Triggers in the foreground dataset marked as false positive can originate from windows where a GW injection is present with a t_c outside the [11.0, 11.2] s range. The behavior of the ML model for these windows is not guaranteed to resemble that of a pure noise sample or a noisy signal sample, and, thus, the false positive label is not accurate. Since we cannot confidently assign the false positive label to these samples, it is not possible to include all foreground dataset triggers for the computation of the ROC curve.

All tests were run on 16 CPU cores and an NVIDIA Quadro RTX 8000 GPU. Each epoch of training takes ≈2 h, excluding validation. Running Sage on evaluation mode, it takes ≈16 h to generate triggers on 1 month of testing data.

A. Comparison with matched filtering

We compare Sage to the PyCBC submission in MLGWSC-1 [21] for the realistic D4 injection study. The submission is based on the standard configuration of PyCBC used for archival searches of compact binary mergers, as seen in [113] (particularly the "broad" configuration *without* an extra tuning term in the ranking statistic for a specific signal population). The template bank they used to evaluate this injection study included templates that account for spins aligned with the orbital angular momentum, did not include precession or higher-order modes, and had mass parameters that were bound by the testing distributions provided in Table I and were placed in a stochastic manner. They account for the presence of glitches [114–116] using waveform consistency checks [39,42] and ensure that the amplitude, phases, and time of arrival in each detector are consistent with the astrophysical population [117]. PyCBC was run separately on the "background" and "foreground" datasets, and a ranking statistic was assigned to potential candidates based on an empirically measured noise distribution and the consistency with the expected GW [117] (without attempting to optimise sensitivity for a particular signal population).

Although PyCBC uses a template bank that does not account for precession or higher-order modes, we do not expect the detection efficiency to be too different from a template bank that does, since neither of those effects is detectable in this injection study (refer to Fig. 1). Although there would exist slight differences in the signal morphology between injections generated using IMRPhenomXPHM and the relatively simpler templates used by PyCBC, the network SNR of each detected injection should not vary significantly due to the simpler templates. Thus, although we use IMRPhenomPv2 to train Sage, there is no precedent to expect a better detection efficiency from Sage solely due to the usage of a waveform model that is relatively more similar to the injections.

1. Injection study

In this section, we describe each of the runs in Table II, compare them, and describe how ML biases influence their detection performance. All aspects of the methodology are consistent with Sec. II, and any changes or choices made are described within each run description. Some choices are common for all runs and are described here.

For all runs, we rescale the optimal network SNR of all samples during training, as mentioned in Sec. II B.

[6]For a fair comparison to the MLGWSC-1 benchmark results, we use the code provided at [112] to evaluate our model.

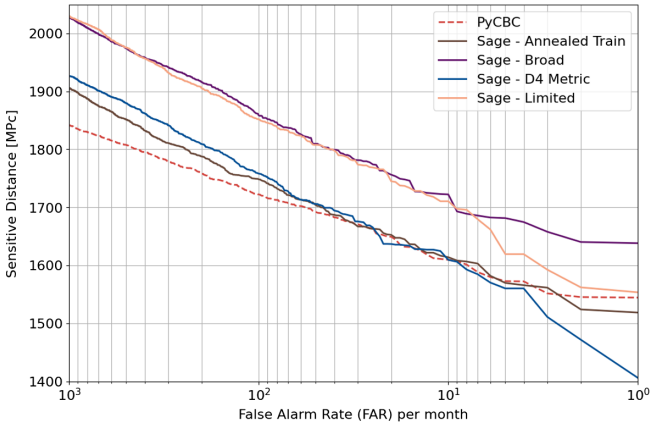


FIG. 9. The sensitive distance metric as a function of false-alarm rate for PyCBC and a set of different Sage analyses of the testing dataset.

From experimentation, we found that using a half-normal distribution allowed for a higher detection efficiency than using a uniform distribution in the same range. The half-normal is described by the probability density function

$$f(x) = \sqrt{\frac{2}{\pi}} \exp \frac{-x^2}{2}, \tag{11}$$

where $x \geq 0$. The samples generated using this density are scaled by a factor of 4 and shifted by a factor of 5. This places the lower bound at an SNR of 5. From a sample size of 10^6, we find that 99% of the distribution lies below an SNR of ≈15.3 and the maximum SNR is ≈24.3.

Sage—Limited. We use detector noise from O3a(day 36–51) and O3b(day 1–113) for training. This is then recolored using precomputed PSDs from the same data with a probability of 0.4 (value chosen via fine-tuning). Figure 9 shows a comparison of the sensitive distances of Sage—Limited and PyCBC as a function of FAR. We see that our run achieves a higher sensitive distance than PyCBC at all FARs from 1000 per month to 1 per month. However, Fig. 10 (top left) shows that the run does not find as many signals as PyCBC. The higher sensitive distance is due to finding a few more high chirp mass signals than PyCBC. As described in Sec. I C, this is a consequence of a bias against low chirp mass signals (incurred from biases due to insufficient information, limited sample representation, sample difficulty, and spectral bias) and a parameter-dependent weighting of the sensitive distance metric. Although our bias mitigation strategies improved the detection efficiency at lower chirp masses compared to runs without them, it is still not sufficient to match PyCBC's detection efficiency.

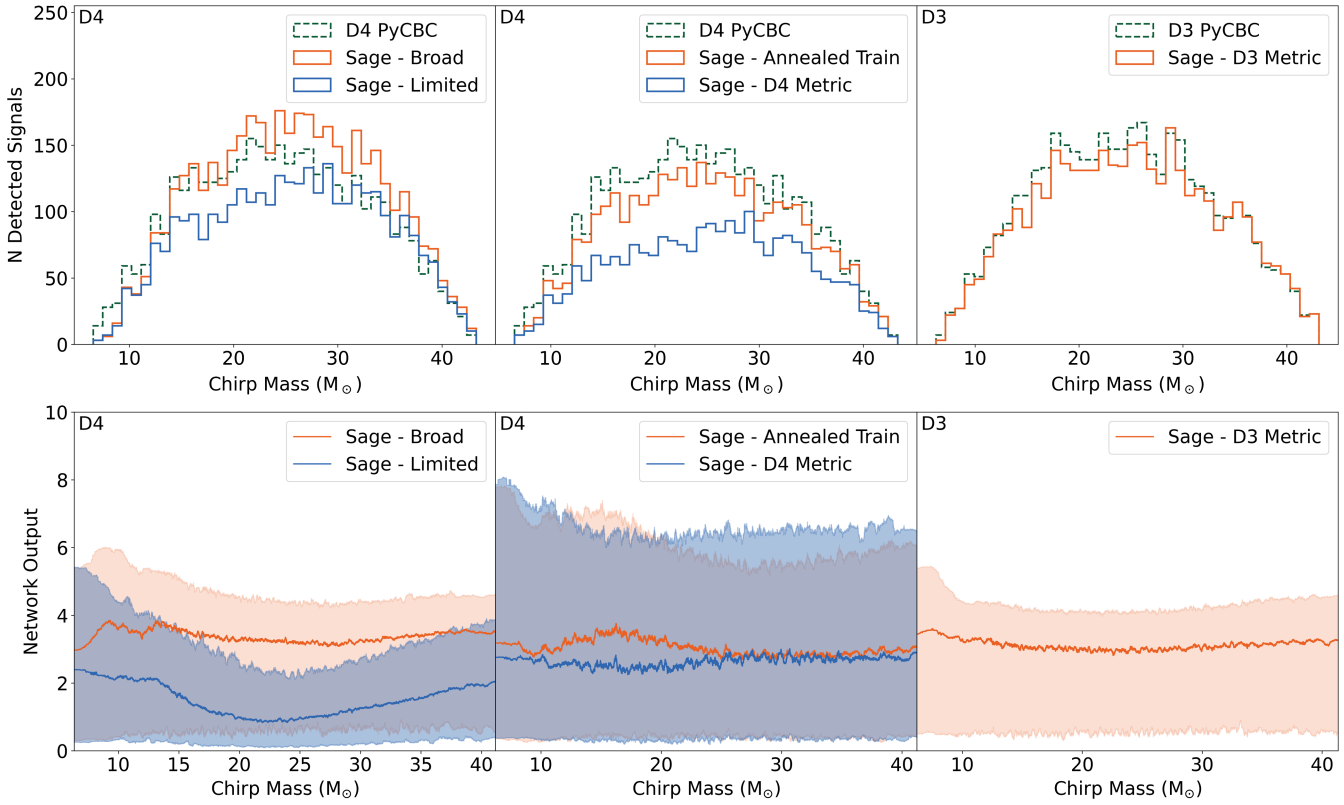


FIG. 10. Top: 1D histograms of detected signals along the chirp mass dimension of different Sage runs with PyCBC at an FAR of 1 per month. The bottom row shows the ranking statistic output by the network compared to the chirp mass $\mathcal{M}_c$ of 250 000 signals. The ranking statistic is put through a running mean across τ where we use a window size of 500 samples and a step size of one sample. The error bar on each plot indicates the region between the 5th and 95th percentile region of the network output at a given chirp mass.

Sage—Broad. The presence of OOD PSDs in the 30-day D4 testing dataset, as seen in Fig. 3, might lead to reduced detection efficiency due to the model's inability to extrapolate to unseen PSDs. In this experiment, we determine whether closing this gap would affect the detection efficiency. The only change we introduce in this experiment is to use O3a(day 1–30) to precompute the PSDs used to recolor the training data O3a(day 36–51) and O3b(day 1–113), instead of using the training data itself. This intentionally introduces some mild overlap between the training and testing datasets, although the recolored noise realizations are, strictly speaking, different from the testing dataset.

From Fig. 9, it is immediately evident that we see a significant boost in the sensitive distance, from an FAR of 10 per month to 1 per month. We see an increase of ≈5.5% and ≈6.5% in sensitive distance when compared to Sage—Limited and PyCBC, respectively, at an FAR of 1 per month. Figure 11 shows the median sensitive distance for this run with an error bar made by repeating this run 3 times with different seeds. The seeds were set such that both signal and noise realizations for all epochs during training are different but within the same distribution and that the networks have different random initializations for their trainable parameters. We also provide the parameter-independent ROC curve that compares this result to PyCBC in Fig. 12. At a false-alarm probability of 10^{-5}, we see an ≈38% increase in the true-alarm probability. Figure 10 (top left) also shows the 1D chirp mass histograms of the detected signals, comparing Sage—Limited and Sage—Broad. The two runs perform similarly at chirp masses below $\approx 10 M_{\odot}$, but the latter outperforms the former significantly everywhere else.

The increase in detection performance when compared to Sage—Limited points to two possible causes: (i) an increase in the variance of training noise realizations

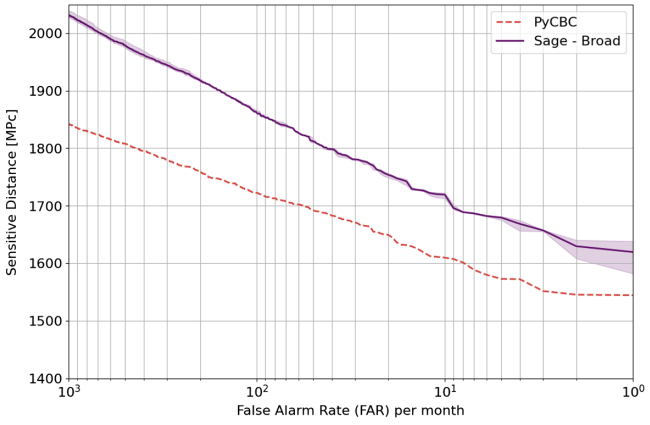


FIG. 11. The sensitive distance metric as a function of FAR for PyCBC and the Sage—Broad run on the testing dataset. The solid violet line is the mean sensitive distance of three runs made with the broad configuration, and the error bar represents the max and min values obtained from the runs.

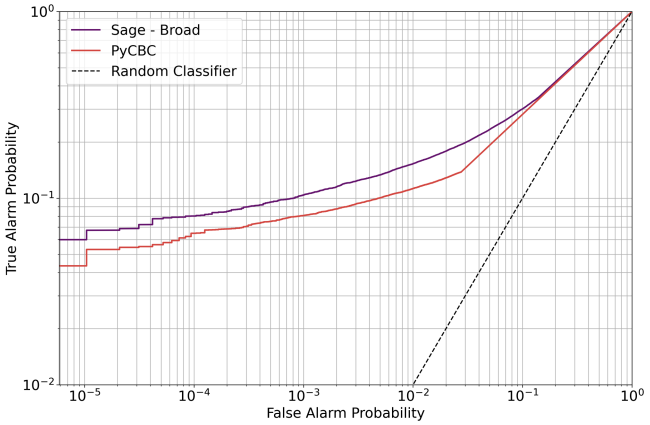


FIG. 12. The true-alarm probability as a function of false-alarm probability (ROC curves) for PyCBC and the Sage—Broad run that achieved the highest sensitive distance on the testing dataset.

(via their PSD) or (ii) all testing noise PSDs are in distribution for Sage—Broad, but the same is not true for Sage—Limited. We will investigate the root cause in the following runs and in Sec. III A 4.

Caveats of the evaluation metrics. We generally attribute an increase in the true-alarm probability of an ROC curve at a given false-alarm probability and the sensitive distance metric at a given FAR to better detection performance. However, this is not entirely correct, and both evaluation metrics have their disadvantages. Assuming Gaussian noise, a model that is biased against low chirp mass signals will operate on a narrower parameter space and, thus, detect more high chirp mass signals at any given FAR. This bias would result in a significant increase in the sensitive distance metric, given the chirp mass dependence of the sensitive volume [see Eq. (5)]. A parameter-independent metric, like the ROC curve, would still account for only the number of detections made and not the distribution of the detections, thus masking any biases. The use of only a single figure of merit based on a predetermined set of signals may be of very limited application to the real-world problem of detecting signals with an unknown parameter distribution. For instance, an algorithm may achieve a high figure of merit while being almost completely insensitive to signals in some region of parameter space, which may be much more prevalent in reality than in the evaluation set. In other words, optimizing detection for a known target population may imply a greater risk of losing sensitivity for the unknown real population.

We visualize the 1D and 2D distributions of the detected signals as histograms in Fig. 13, to compare and visualize any differences between Sage—Broad and PyCBC. We find that 3344 signals are detected by both pipelines, 513 signals were found only by PyCBC, and 946 signals were found only by Sage at an FAR of 1 per month. This is equivalent to an increase of ≈11.2% in the number of signals detected compared to PyCBC in the MLGWSC-1

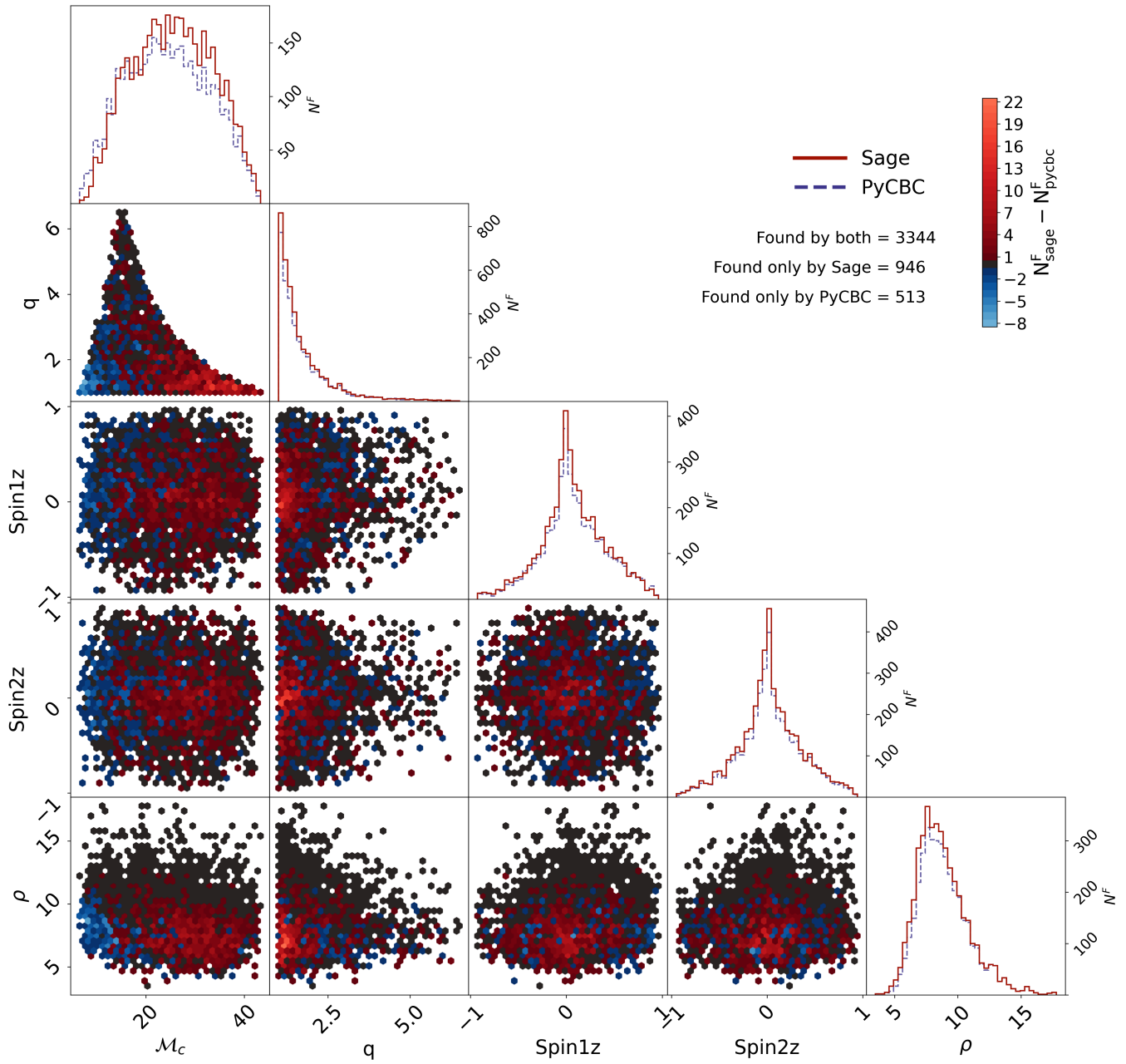


FIG. 13. 1D and 2D histograms of detected signals for different GW parameters at an FAR of 1 per month, comparing the best Sage—Broad run and PyCBC. We chose GW parameters where we expected to see learning bias. The diagonal subplots contain 1D histograms of the detected signals for a given parameter. The 2D histograms are hexagonal binned plots, where each bin is colored using $N^F_{\text{sage}} - N^F_{\text{pycbc}}$, where N^F_{sage} and N^F_{pycbc} are the number of signals detected by Sage and PyCBC, respectively. We use a divergent color map that is set to be black when $N^F_{\text{sage}} - N^F_{\text{pycbc}}$ is zero, is progressively brighter in red along the positive axis, and is progressively brighter in blue along the negative axis. So, red and blue patches can be attributed to regions where Sage and PyCBC detect more signals, respectively.

realistic injection study. We find no anomalous biases present across any parameter that is not directly related to chirp mass, mass ratio, or SNR. A full set of 1D histograms for all relevant intrinsic and extrinsic parameters is provided as 1D histograms in Fig. 22 in Appendix B.

Important note. In Fig. 13, Sage—Broad and PyCBC operate on two different mass distributions—the former on $U(m_1, m_2)$ and the latter on $U(\tau_0, \tau_3)$. For a fair comparison of their detection sensitivities, the search statistic for PyCBC must be fine-tuned and optimized for the $U(m_1, m_2)$ mass distribution (see [27,28] for relevant information). This fine-tuning will likely lead to an increase in the number of signals detected by PyCBC in the high chirp mass region. However, given that the effect of blip

glitch noise is much higher in this region, it is not straightforward to speculate on the extent of improvement that might be seen in a fine-tuned PyCBC [42,118]. This study is left for future work. Furthermore, the small number of low-SNR, low chirp mass signals missed by Sage in Fig. 13 could be attributed to this difference in mass distribution.

Sage—D3 metric. We found many training training distribution configurations that reduced the bias across the chirp mass and mass ratio dimensions, but for this run, we investigate using $U(\tau_0, \tau_3)$ to set the training distribution for the component masses. A detailed description for using this distribution for training is defined in Sec. II B. The key difference between $U(m_1, m_2)$ (which we used in the previous runs) and $U(\tau_0, \tau_3)$ is that the latter has an underdensity of training data samples in the high chirp mass region (see Fig. 4). For this run, we train and test on the D3 dataset setting with simulated colored Gaussian noise instead of real noise. This setup is ideal for studying the effects of varying the training distributions of GW parameters, as it avoids introducing learning biases from real detector noise.

Figure 10 (top right) compares the 1D histograms of the detected signals at an FAR of 1 per month of Sage—D3 metric to PyCBC. We see that using $U(\tau_0, \tau_3)$ as the training mass distribution alongside simulated Gaussian noise produces a detection efficiency very similar to PyCBC at all chirp masses. This justifies the use of the parameter distribution on real detector noise as well.

Sage—D4 metric. This run also uses $U(\tau_0, \tau_3)$ to set the training distribution for component masses but uses real detector noise (similar to all other runs on D4) instead of simulated colored Gaussian noise. Compared to Sage—D3 metric, we see that Sage—D4 metric has significantly lower detection efficiency, in Fig. 10 (top middle). This reduction is solely due to the attributes of real noise; it is either due to the inability to properly reject non-Gaussian artifacts or due to the difference between the training and testing noise PSDs.

Sage—Annealed train. An underdensity of training data samples in the high chirp mass region results in worse noise rejection capability, as seen in Sage—D4 metric, and an overdensity results in a bias against long-duration signals, as seen in Sage—Broad. We tackle this issue by viewing the GW detection problem as the learning of two independent tasks: noise rejection capability and unbiased learning of the GW parameter space. Since the learning of these tasks simultaneously is not trivial, we introduce the concept of annealed training. For the Sage—Annealed train run, we train a network similar to Sage—Broad for 20 epochs [with a mass distribution given by $U(m_1, m_2)$] and then transition the mass distribution to $U(\tau_0, \tau_3)$ over the course of the next 20 epochs. During the transition phase, the ratio between the number of samples generated using $U(\tau_0, \tau_3)$ to $U(m_1, m_2)$ varies as a function of epoch.

The intent of annealing is twofold: taking advantage of catastrophic forgetting [119] and the concept of "unforgettable" examples [120,121]. In the context of continual learning, catastrophic forgetting is the tendency of a neural network to abruptly lose knowledge relevant to a previous task as information about the current task is incorporated [119]. This is undesirable in continual learning but beneficial to our problem for shifting the training distribution from $U(m_1, m_2)$ to $U(\tau_0, \tau_3)$. Unforgettable examples (or, more precisely, difficult-to-forget examples) are input data samples that are learned early during training and typically have low losses during the bulk of the training [121]. This allows for all high-SNR signals to be remembered well throughout training and not be forgotten during the annealing process. This should allow for any noise rejection capability learned via high-SNR signal features to also be retained.

Figure 10 (top middle) compares the 1D chirp mass histograms of the detected signals at an FAR of 1 per month between the Sage—Annealed train and Sage—D4 metric runs. We see a significant increase in the detection efficiency of the former run compared to the latter, proving the efficacy of the annealing training strategy.

Network output behavior. We evaluate all the above runs on a validation dataset of 500 000 samples (1:1 ratio between signal and noise classes) comprised of signals and noise from the testing dataset distribution. Figure 10 (bottom) shows the network output obtained from this validation dataset for the signal class as a function of its corresponding chirp mass. The larger the network output, the more confident the model is to place that signal in the signal class. Notable observations have been described below.

Observation 1. There is a significant increase in the confidence of Sage—Broad when classifying signals across the whole range of chirp masses when compared to Sage—Limited. This is especially evident above a chirp mass of $\approx 15 M_\odot$. Sage—Broad has the least classification confidence for chirp masses below $\approx 10 M_\odot$ which contributes to the lower number of detections in this chirp mass regime compared to PyCBC.

Observation 2. The annealing procedure for Sage—Annealed train improves the classification confidence from a chirp mass of $\approx 10 M_\odot$ till $\approx 25 M_\odot$. This indicates that the increase in detection efficiency seen in Fig. 10 (top middle) for Sage—Annealed train over Sage—D4 metric is primarily due to retaining information in this chirp mass region when annealed from the $U(m_1, m_2)$ mass distribution. Surprisingly, signals with a chirp mass above $\approx 25 M_\odot$ do not contribute much to the improved noise rejection capability.

Observation 3. On average, the Sage—D3 metric network is a lot more confident with classification than Sage—D4 metric throughout the chirp mass range. Although the latter, on average, has lower confidence, it detects many signals across the whole range of chirp mass with higher

confidence than Sage—D3 metric and even Sage—Broad. Using the metric as the training distribution leads to a lower *bias due to limited sample representation*, and, in turn, the network is provided with more GW domain knowledge. This allows it to be more confident with some signals. However, Sage—Annealed train and Sage—D4 metric do not seem to perform well at lower SNRs, leading to the worse average performance (there are more low-SNR than high-SNR signals; see the SNR distribution in Fig. 21 in Appendix A).

Observation 4. Interestingly, the upper error bar on all runs shows that the networks find longer signals with the highest-ranking statistics. However, other than Sage—D3 metric, none of the other runs are good at low-SNR, low chirp mass signals, leading to a lower average network output at low chirp masses.

2. Learning dynamics

Here, we investigate the difference between Sage—Broad and Sage—Annealed train to potentially bridge the gap between the two runs. We create four auxiliary validation sets to analyze the learning dynamics in different regions of the parameter space. We chose all unique combinations without duplicates, between the two sets of distributions: $\{U[\mathcal{M}_c^{\rm low}, \mathcal{M}_c^{\rm median}], U[\mathcal{M}_c^{\rm median}, \mathcal{M}_c^{\rm high}]\}$ and $\{\delta(\mathrm{SNR}=5), \delta(\mathrm{SNR}=12)\}$, where δ is the Dirac delta functions. This creates four datasets with different chirp mass and SNR regimes. At the end of each training epoch, we freeze the weights and run the network on each of these auxiliary validation datasets with 500 000 samples (1:1 ratio between signal and noise classes). This provides us with the learning dynamics of the network in each regime. Figure 14 depicts these four losses (obtained only for the signal class) over the course of 40 epochs of training for the runs Sage—Broad and Sage—Annealed train.

Low-SNR regime [see Fig. 14 (top)]. For the Sage—Broad run, we see that low chirp mass signals are more difficult to learn compared to high chirp mass signals, given that they have higher losses. It is interesting that both losses increase with more training, contradictory to typical loss curve dynamics. This is likely due to the need to reject glitches which could be similar in morphology compared to these signals.

High-SNR regime [see Fig. 14 (bottom)]. We see a surprising trend where low chirp mass signals have much lower losses compared to high chirp mass signals, despite the expected biases against low chirp mass signals. Both losses have a decreasing trend, which is consistent with expectations, since they should be easier to distinguish from non-Gaussian noise given their SNR.

Effect of annealing. The transition from $U(m_1, m_2)$ to $U(\tau_0, \tau_3)$ happens at epoch 20. Here, we see a mild increase in the loss for high-SNR, low chirp mass signals despite having an overdensity of low chirp mass signals, which is indicative of a contribution to noise rejection capability.

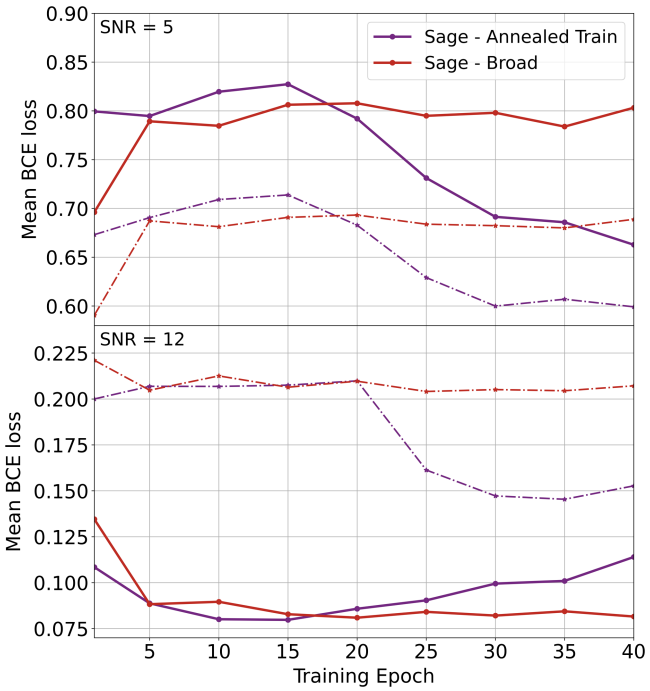


FIG. 14. Mean BCE loss as a function of training epochs (only for the signal class). The plots compare the auxiliary dataset validation loss curves for the Sage—Broad and Sage—Annealed train runs. The solid lines correspond to the low chirp mass dataset $U[\mathcal{M}_c^{\rm low}, \mathcal{M}_c^{\rm median}]$, and the dashed lines correspond to the high chirp mass dataset $U[\mathcal{M}_c^{\rm median}, \mathcal{M}_c^{\rm high}]$. As the plot text states, the top plot is in the low-SNR regime (= 5), and the bottom row is in the high-SNR regime (= 12).

We also observe that the losses pertaining to high chirp mass signals decrease steadily during the annealing procedure at both SNR regimes. However, despite having lower losses than Sage—Broad at high chirp masses, Sage—Annealed train is not as performant. This is because we do not show the total loss function, which includes the noise class as well. The total loss of Sage—Annealed train is higher than Sage—Broad, which explains the discrepancy.

Given the overall lower losses for the signal class in Sage—Annealed train compared to Sage—Broad, the key to improving the detection efficiency of Sage—Annealed train must lie in improving the noise rejection capability. This requires an in-depth investigation of the noise behavior and its effects in the learning of different regions of the GW parameter space. We leave this study for the future.

3. Glitch rejection capability

In this section, we investigate the effect of the presence of glitches in the testing dataset. If the highest noise triggers are caused by glitches, the model would be extremely brittle to the testing dataset noise realizations and its detection efficiency at lower FARs has the potential to vary significantly with different datasets.

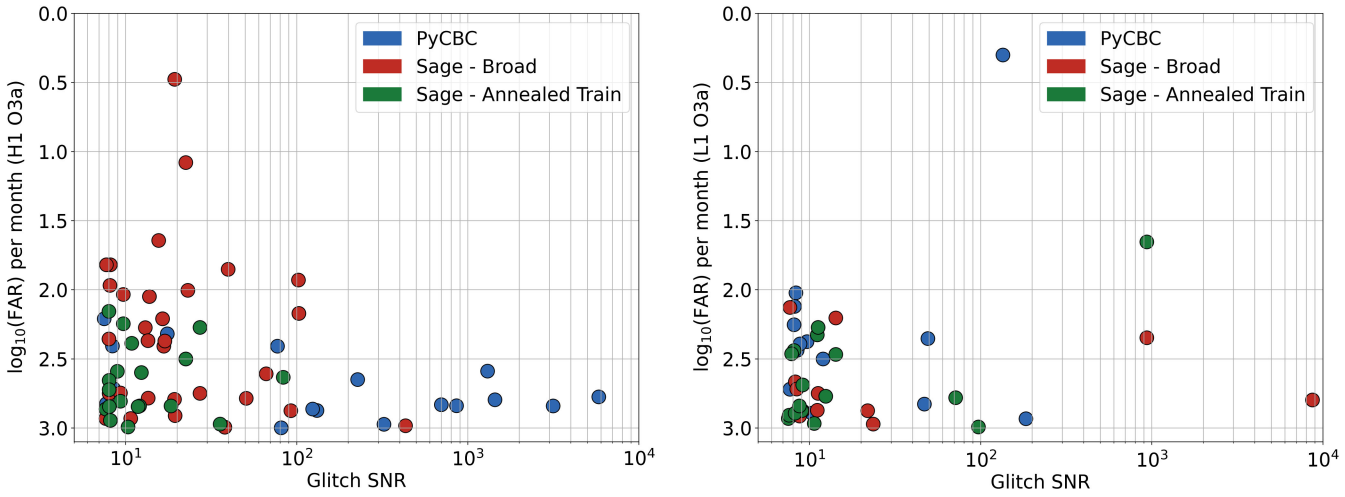


FIG. 15. Log FAR per month as a function of glitch SNR (obtained from GravitySpy [82]) for the H1 and L1 detectors. The scatter plot shows the glitch triggers produced by different Sage runs and PyCBC from the two detectors in the O3a testing dataset noise.

We get the start time and duration of all glitches reported in the Omicron trigger pipeline [122], with a glitch SNR above 7.5 and a peak frequency of the noise event between 10 and 2048 Hz, for the O3a testing dataset noise segments in the H1 and L1 detectors. The triggers that lie within the start GPS time and end GPS time of each glitch are obtained from Sage for different experiments and PyCBC for the D4 submission to MLGWSC-1. We then maximize these triggers to obtain the ranking statistic of a single trigger that represents each glitch. Omicron produces a glitch SNR from the time-frequency representation of the strain and cannot be compared to the integrated signal SNR. We plot the maximized ranking statistic of triggers below an FAR of 1000 per month against its corresponding glitch SNR in Fig. 15 for the runs Sage—Annealed train and Sage—Broad and compare it with PyCBC.

We observe that all triggers produced by Sage—Annealed train in H1 and L1 are above an FAR of 100 per month and 1 per day, respectively. Sage—Broad consistently seemed to produce a lot more triggers comparatively and at FARs below the lowest FAR observed in the Sage—Annealed train for the H1 detector. However, both runs seem to perform similarly well in L1, with the annealed run producing the trigger with the lowest FAR. Although we expect the annealed run to have better glitch rejection capability (due to lower *bias due to limited sample representation* and, in turn, higher GW domain knowledge), we cannot conclude much from this comparison, since this is a relatively small testing dataset. However, comparing our results with PyCBC, we can conclude that the performance is similar, except for PyCBC producing a trigger at an FAR of 2 per month in L1.

We investigate a little further into the comparison of generated triggers for different types of glitches in the appendixes (see Fig. 23 in Appendix C).

4. Effects of nonstationary noise

Since Sage and PyCBC seem to have similar glitch rejection capability, the primary cause for performance degradation in some Sage runs must be due to the differences in noise PSDs between the training and testing dataset. We investigate this possibility by comparing the results between three runs: Sage—D3 metric, Sage—Limited, and Sage—Broad. With this investigation, we test three hypotheses: (i) If there is no difference between the training and testing noise PSD distribution, the model should perform identically in both settings, (ii) a model trained with a broader noise PSD distribution will be more confident in predicting the presence of a GW signal compared to a model with a narrower distribution, and (iii) if the testing noise PSD distribution is out of distribution compared to the training noise PSD distribution, the model might not be able to extrapolate to the unknown PSDs.

We create four different testing datasets (also known as a *test set* within the plots), each with 500 000 samples of noise for studying the effects of time-varying PSDs on Sage. These datasets are described below.

Datasets 1 and 2. The first and second testing datasets contain simulated coloured Gaussian noise with PSDs from the same distribution as that described for the Sage—D3 metric run. The seed for the training and testing datasets for Sage—D3 metric runs determines the random choice of PSDs from the precomputed set when generating the noise samples. The first testing dataset uses the same seed as the training dataset, and the second does not.

Datasets 3 and 4. The third and fourth datasets contain noise samples retrieved from O3a(day $36-51$) + O3b(day $1-113$) and O3a(day 1–30) segments, respectively. The inclusion or omission of O3a(day 1–30) from the training or testing datasets significantly affects the variance of the noise PSDs as seen in Fig. 3, where the yellow (in H1) and purple (in L1) PSDs lie within this

TABLE III. Parameters used to produce the GW signal used in Sec. III A 4. The SNR is rescaled to the required value, which dictates the rescaled distance parameter.

Parameter	Value
Approximant	IMRPhenomPv2
Mass 1	34.79
Mass 2	26.21
Right ascension	3.11
Declination	−0.62
Inclination	0.83
Coalescence phase	1.75
Polarization	2.88
Time of coalescence	11.09
Spin1x	−0.30
Spin1y	0.11
Spin1z	0.01
Spin2x	0.28
Spin2y	−0.25
Spin2z	−0.09
Chirp mass	26.24
Mass ratio	1.33

segment. The seed for these two datasets determines the random start time for each noise sample. The start time is uniformly distributed along the segments provided. The testing seed is not equal to the training seed for either of the datasets.

We then generate a single arbitrarily chosen GW signal with parameters provided in Table III and rescale its network optimal SNR across H1 and L1 to be a fixed value of 10. This signal is injected at the provided time of coalescence into half of the noise samples in each dataset, producing a 1∶1 ratio between the signal and noise classes. A summary of the different runs made is provided in Table IV. The rest of this section is dedicated to the analysis of Fig. 16.

General description of Fig. 16. After evaluating a particular testing dataset with a Sage run, we histogram the network output (or ranking statistic) produced by the model for the signal and noise class. The plots in the left column correspond to the noise class and the ones in the right to the signal class. The histograms should resemble a noncentral chi-squared distribution for both classes given the behavior of real noise PSDs. This is true for the signal class, as they all contain the same signal with a fixed network optimal SNR and GW parameters. The intersection of the tails of the two classes contributes to type-I and type-II errors in the detection problem. The classification confidence of a model for a given dataset can be associated with the interclass distance (distance between average network outputs of the classes) and the reduction in type-I and type-II errors.

Row 1 in Fig. 16. The Sage—D3 metric is evaluated on the first two testing datasets to produce the first row of results in Fig. 16. We see that the model produces similar results for both testing datasets (1 and 2). This corroborates the hypothesis that the model should perform identically in the training and testing settings, given that the PSD distribution is identical (ignoring non-Gaussian noise).

Row 2 in Fig. 16. Here, we test Sage—Limited and Sage—Broad on the third dataset, with noise retrieved from $\mathrm{O3a}(\mathrm{day}\ 36-51)+\mathrm{O3b}(\mathrm{day}\ 1-113)$. Our naive expectation might be that Sage—Limited, which is trained on the same noise segment, should perform better than Sage—Broad as it is trained on precisely the required PSD realizations. However, we observe that Sage—Broad is more confident in classifying a signal injected into the same noise realizations than Sage—Limited, corroborated by the larger interclass distance. Moreover, Sage—Broad also seems to perform slightly better when classifying pure noise realizations from this dataset as seen in the left plot. This corroborates the hypothesis that a model trained with a broader noise PSD distribution will be more confident in predicting the presence of a GW signal compared to a model with a narrower distribution.

Reason for classification confidence. This might not be simply due to a broader PSD distribution but rather the inclusion of more challenging noise PSD realizations.

TABLE IV. Summary of run configurations used to make Fig. 16. The first column points to a histogram in a particular row in Fig. 16. The second column provides the name of the Sage run used to generate the histogram (details about the runs can be obtained from Sec. III A 1). The third column describes the PSD distribution used to generate the training dataset samples. Sage—D3 metric uses a fixed set of O3 PSDs as described in Sec. II C. Sage—Broad and Sage—Limited both use noise samples from $\mathrm{O3a}(\mathrm{day}\ 36-51)+\mathrm{O3b}(\mathrm{day}\ 1-113)$ for training, but the former uses PSDs from O3a(day 1–30) to recolor the training samples, thus broadening the training PSD distribution. The last column describes the PSD distribution used to generate the testing dataset samples.

Pointer	Model	Train PSD distribution	Test PSD distribution
Row 1 red	Sage—D3 metric	Fixed set of O3 PSDs	The same as train
Row 1 blue	Sage—D3 Metric	Fixed set of O3 PSDs	
Row 2 orange	Sage—Limited	$\mathrm{O3a}(\mathrm{day}\ 36-51)+\mathrm{O3b}(\mathrm{day}\ 1-113)$	The same as Sage—Limited
Row 2 purple	Sage—Broad	$\mathrm{O3a}(\mathrm{day}\ 1-51)+\mathrm{O3b}(\mathrm{day}\ 1-113)$	
Row 3 pink	Sage—Limited	$\mathrm{O3a}(\mathrm{day}\ 36-51)+\mathrm{O3b}(\mathrm{day}\ 1-113)$	O3a(day 1–30)
Row 3 green	Sage—Broad	$\mathrm{O3a}(\mathrm{day}\ 1-51)+\mathrm{O3b}(\mathrm{day}\ 1-113)$	

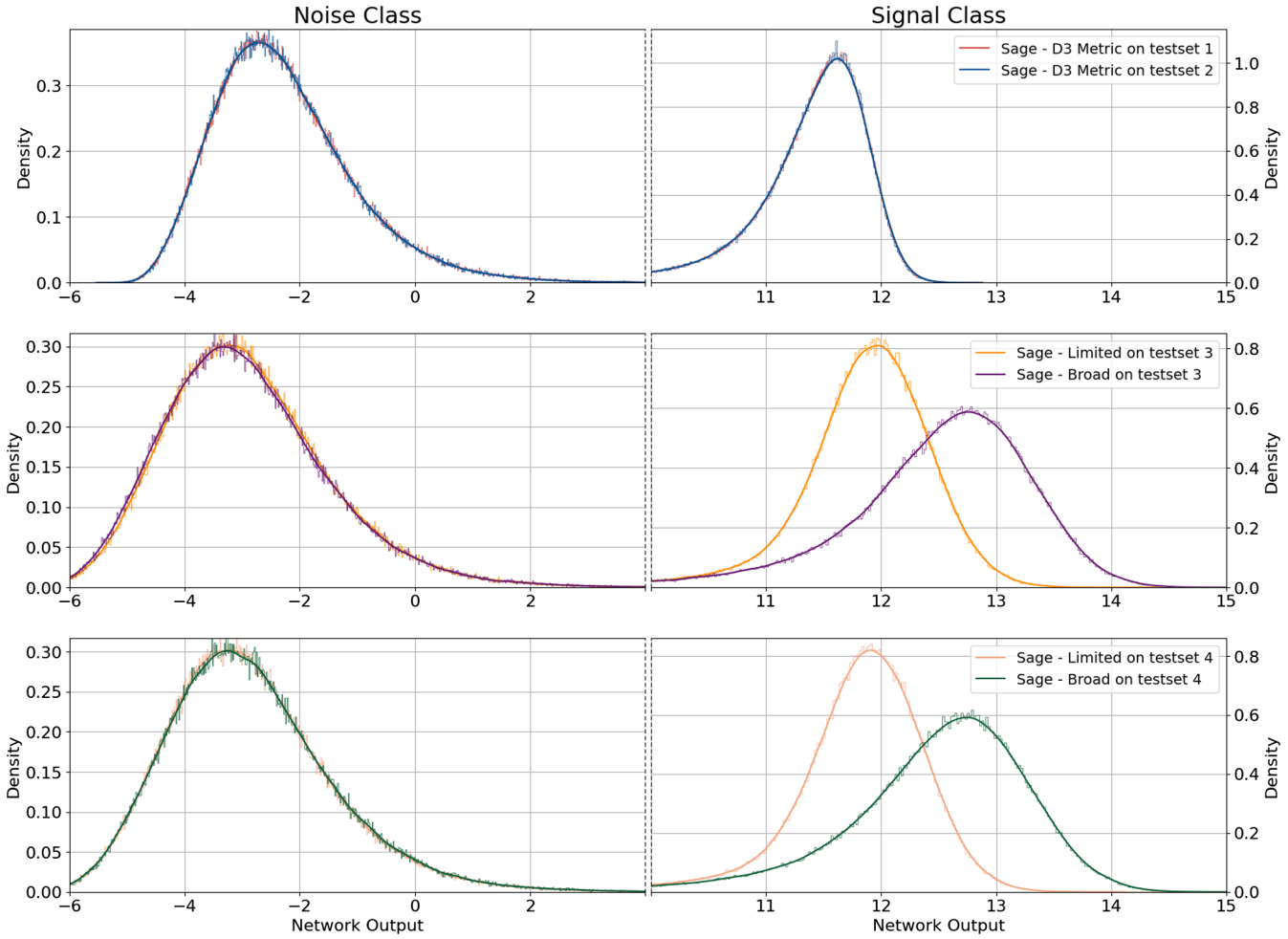


FIG. 16. Normalized histograms of ranking statistics generated by Sage runs for samples from the signal and noise class on a particular testing dataset of 500 000 samples (1 : 1 ratio between the signal and noise class). A kernel density estimate of each histogram is plotted over each histogram. The left column corresponds to the noise class, and the right column corresponds to the signal class. We truncate both histograms in the region of overlap between the two classes for plotting purposes.

Resolving the sources of type-I and type-II errors might lead to more confident models. The outcome of better generalization due to the inclusion of harder training samples has been observed in many previous works [123–125] and is, thus, a reasonable conclusion. However, we also found that simply decreasing the lower bound of the training SNR distribution did not yield the same results. Thus, the improvement in confidence is likely due to better generalization on noise PSDs due to addressing the *bias due to lack of variation*.

Row 3 in Fig. 16. We repeat the same procedure as row 2, but using the fourth dataset, with noise retrieved from O3a (day 1–30). The behavior of the two networks is similar to that seen in row 2, with the exception that both networks are now equally confident about the noise class outputs. Although Sage—Broad performed better than Sage—Limited in testing dataset 3 for the noise class, this behavior did not transfer to PSDs in testing dataset 4. The similar performance of the two models on the noise class indicates that Sage—Limited is able to successfully extrapolate to unknown PSDs with higher power. The unchanged behavior with the signal class in row 2 and row 3 signifies that neither network is overfitting.

Is real noise better than Gaussian noise for training? Comparing row 1 to rows 2 and 3, we see that models trained on real noise (Sage—Limited and Broad) have higher interclass distance than the model trained on simulated colored Gaussian noise (Sage—D3 metric). This is due to two reasons: (i) The precomputed PSDs used for Sage—D3 metric does not vary significantly, thus leading to nearly white noise after whitening with the median PSD, and (ii) absence of non-Gaussian noise artifacts. Previous works [126,127] have established that deep networks learn patterns easily and require less network capacity to do so, but white Gaussian noise (or the lack of patterns) is not learned but memorized and takes a large amount of network capacity. Moreover, trying to learn Gaussian noise was shown to lead to complex decision boundaries and required longer training times. Thus, the lack of patterns in noise samples hinders learning.

Implications. Data preprocessing procedures that effectively whiten the noise samples (by using an estimated PSD or glitch removal), counterintuitively, might hinder learning for the GW detection problem. Using a broader training distribution for noise PSD might allow the model to understand and interpolate the PSD space rather than memorize it. Interpolation, in turn, leads to better generalization and reduces the need to use larger training datasets (data efficiency).

B. Comparison with other ML pipelines

In this section, we compare our results with recent ML pipelines that have evaluated their work on the datasets provided in MLGWSC-1 [21]. We do a more in-depth comparison with AresGW [53] and Aframe [56] given that they exhibit the highest detection sensitivity among the pipelines considered.

1. Comparison with AresGW

Data and architecture differences between AresGW and Sage. The AresGW architecture is a 1D ResNet-54 trained using the curriculum learning strategy [53,128]. For training, they use a 1-s sample length, 12 days of real O3a noise data, and 38 000 different waveforms (each injected into 19 different background segments, leading to 740 000 different training samples). They use the same training mass distribution to generate waveforms as Sage—Broad.

Figure 17 shows the distribution of detections across different parameters and compares the number of detections between Sage—Broad and AresGW at an FAR of 1 per month. We see that 2814 signals were found by both pipelines, 1476 signals were found only by Sage, and 79 signals were found only by AresGW. This is equivalent to an increase of ≈48.29% in the number of signals detected at that FAR. Sage—Broad particularly outperforms its counterpart in the lower chirp mass regime.

The AresGW pipeline was enhanced in [54] and included the following major changes: using 35 days of real O3a noise instead of 12, a low-pass filter at 350 Hz for all samples, and changes to how the triggers are generated and aggregated during testing. The relative difference in detection sensitivity between Sage and enhanced-AresGW should not change by much due to the additional noise data, since the latter does not mitigate the biases in Sec. I C. This statement holds even if enhanced-AresGW were trained with 160 days of real O3 noise, 80 million training samples with a length of 12 s, and the same training strategies as Sage. Justification for these statements is provided with sufficient detail in ablation studies II and III (see Sec. III C).

2. Comparison with Aframe

Data and architecture differences between Aframe and Sage. The Aframe architecture is a 1D ResNet-54 trained using the curriculum learning strategy [56,128]. For training, they use a 1.5-s. sample length, ≈4.5 days of real O3 noise data, and 75 000 different waveforms (each injected into a random noise segment obtained from the ≈4.5 days of real noise). A key area of difference that sets apart Aframe from the other pipelines we compare to is the training mass distribution. They use $p(m_1) \propto m_1^{-2.35}$ and $p(m_2|m_1) \propto m_2$ as per GWTC-3 [11], which places a higher emphasis on lower chirp mass or longer-duration signals. Moreover, they train on a mass range between 5 and $100M_\odot$.

Figure 18 shows the distribution of detections across different parameters and compares the number of detections between Sage—Broad and Aframe[7] at an FAR of 1 per month. We see that 2837 signals were found by both pipelines, 1453 signals were found only by Sage, and 71 signals were found only by Aframe. The number of signals detection is similar to AresGW, but the distribution is expectedly different given the different training mass distribution (see Fig. 17 for reference). The 1D histogram of chirp mass in Fig. 18 shows that Aframe addresses the *bias due to limited sample representation* of low chirp mass signals via the training mass distribution, thus detecting more longer-duration signals than AresGW. The 1D histogram of optimal SNR is similar to that of AresGW, indicating a limited capacity to effectively detect low-SNR signals.

For a more meaningful comparison, we must train Sage on the $p(m_1) \propto m_1^{-2.35}$ distribution, which will likely detect more long-duration signals than Sage—Broad, as indicated by the Sage—Annealed train run in Fig. 10. As Aframe uses a 1.5-s sample length, they do not address the *bias due to insufficient information*, and a testing dataset that contains significantly more long-duration signals will enunciate this bias. We leave this analysis to the future.

Consolidated comparison. We consolidate the comparison of the detection performance for different pipelines in Table V and show the sensitive distance metric of multiple ML pipelines and PyCBC at different FARs in Fig. 19. We see an increase of ≈4.1% in sensitive distance compared to AresGW at an FAR of 1 per month. For the sake of completeness, we provide the sensitive distance curve of Aframe in Fig. 19. The disadvantage of using a parameter-dependent metric is clearly evident when comparing our Sage—Annealed train run with AresGW. Although our run finds ≈15.5% more signals at an FAR of 1 per month, it scores ≈3.5% less on the sensitive distance metric, due to the chirp mass dependence.

C. Ablation study on Sage

Although a common practice for pipelines in the ML literature [129], ablation studies are generally not performed in the ML for GW literature. Given the diversity of

[7]Results obtained via private communication.

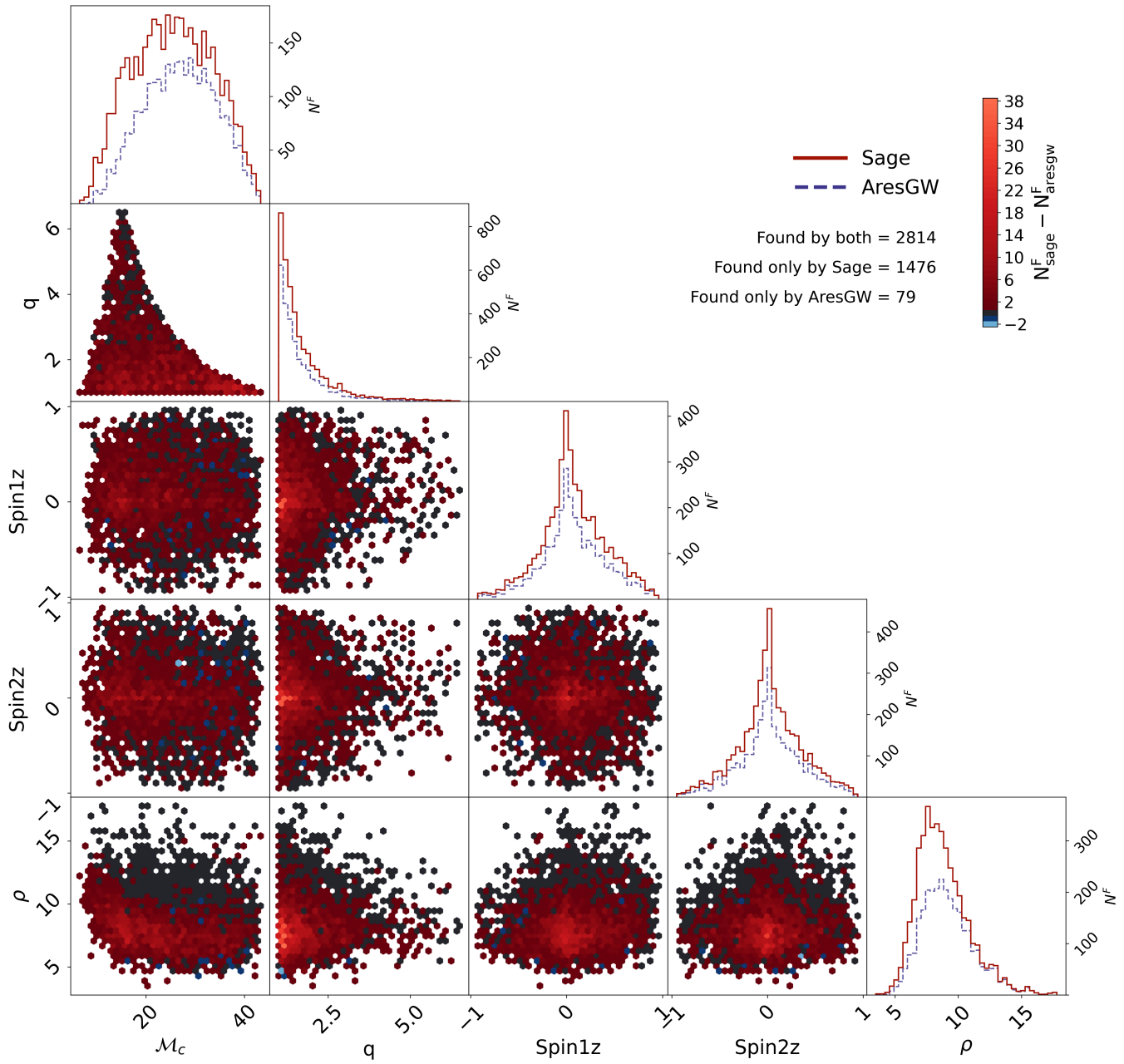


FIG. 17. 1D and 2D histograms of detected signals for different GW parameters at an FAR of 1 per month, comparing the best Sage—Broad run and AresGW. We chose GW parameters where we expected to see learning bias. The diagonal subplots contain 1D histograms of the detected signals for a given parameter. The 2D histograms are hexagonal binned plots, where each bin is colored using $N^F_{\text{sage}} - N^F_{\text{aresgw}}$, where N^F_{sage} and N^F_{aresgw} are the number of signals detected by Sage and AresGW, respectively. We use a divergent color map that is set to be black when $N^F_{\text{sage}} - N^F_{\text{aresgw}}$ is zero, is progressively brighter in red along the positive axis, and is progressively brighter in blue along the negative axis. So, red and blue patches can be attributed to regions where Sage and AresGW detect more signals, respectively.

methods that can be used in ML, it is imperative to prove the necessity of important parts of our methodology.

Although we have shown proof and explanations for the performance gains seen in Sage compared to previous ML pipelines, the source of this performance is not apparent. We try to understand the contribution of some of the major components described in the methodology (see Sec. II) by systematically removing them and evaluating Sage's performance on the same realistic testing dataset. The methodology used throughout the ablation study is the same as that seen in Sec. II, and any changes made are described in each study. The training seeds are fixed for all studies, so they use the same signal and noise realizations for all epochs. The studies that use a limited dataset are trained on

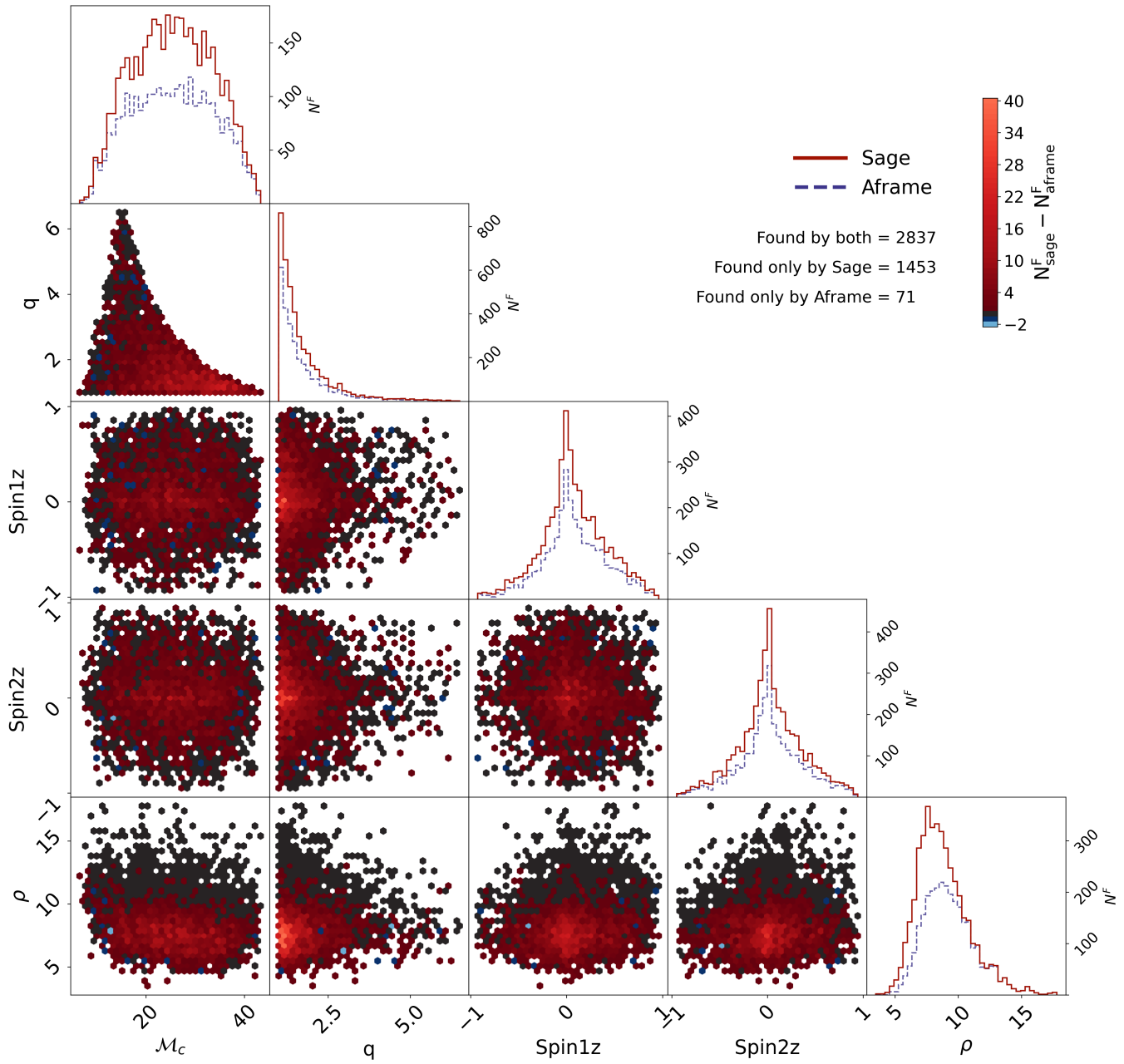


FIG. 18. 1D and 2D histograms of detected signals for different GW parameters at an FAR of 1 per month, comparing the best Sage—Broad run and Aframe. We chose GW parameters where we expected to see learning bias. The diagonal subplots contain 1D histograms of the detected signals for a given parameter. The 2D histograms are hexagonal binned plots, where each bin is colored using $N^F_{\text{sage}} - N^F_{\text{aframe}}$, where N^F_{sage} and N^F_{aframe} are the number of signals detected by Sage and Aframe, respectively. We use a divergent color map that is set to be black when $N^F_{\text{sage}} - N^F_{\text{aframe}}$ is zero, is progressively brighter in red along the positive axis, and is progressively brighter in blue along the negative axis. So, red and blue patches can be attributed to regions where Sage and Aframe detect more signals, respectively.

the same premade dataset. All comparison plots are made at an FAR of 1 per month. This ablation study is not exhaustive and targets only some of the major differences that led to significant performance gains.

Study I—Decreased noise and signal realizations. For the first study, we removed several major components of Sage that were meant to increase the variance in the signal and noise classes during training. For the signal class, we generate the plus and cross polarizations of 10^6 GW signals with the same generation settings as Sage—Broad. Each iteration of every epoch is given a unique seed, which is used to set the right ascension, declination, and polarization for the antenna pattern before calculating the strain of the waveform as measured by the detector. A time shift is

TABLE V. Summary of sensitive distance measures and number of signals detected in the realistic testing dataset for different pipelines and Sage runs, for different FARs. $N^F_{1/\text{month}}$ and $N^F_{10/\text{month}}$ are the number of found injections at an FAR of 1 per month and 10 per month, respectively. The Aframe results are marked with an asterisk (*) as their pipeline is not optimized for the D4 mass distribution in MLGWSC-1 [21].

Pipeline	FAR 100 per month	FAR 10 per month	FAR 1 per month	$N^F_{10/\text{month}}$	$N^F_{1/\text{month}}$
PyCBC [21]	1722.008	1607.307	1538.768	4364	3857
CWB [21]	1364.195	1440.848	1460.617	...	...
Aframe* [56]	1708.237*	1570.450*	1441.291*	4252*	2908*
AresGW [53]	1817.074	1632.455	1573.914	3878	2893
TPI-FSU Jena (2024) [64]	1583.625	1424.370	1316.161	2932	1609
MFCNN [21]	1269.034	984.995	587.930	444	89
Sage—Broad	**1858.554**	**1722.352**	**1638.204**	**4760**	**4290**
Sage—Limited	1851.586	1710.573	1553.479	4731	3306
Sage—D4 metric	1758.124	1609.436	1405.947	4179	2316
Sage—Annealed train	1748.338	1613.982	1518.557	4280	3341

applied for the given detector relative to the assumed geocentric frame to obtain the strain for each of the considered detectors. We then rescale the network SNR of the system by drawing SNR values from a uniform distribution within the bounds [5, 15]. This procedure limits the GW parameter augmentation to the four parameters mentioned above. All other parameters have a fixed 10^6 realizations throughout the training procedure. For the noise class, we retrieve 10^6 samples from O3a(day 31–51) noise segment (with sample overlap) and do not apply any noise recoloring.

Figure 20(a) shows the chirp mass histogram of the number of found injections compared to PyCBC at an FAR of 1 per month. It also shows reference histograms for the PyCBC and best Sage—Broad results on this dataset, taken from Fig. 13. We use only chirp mass here, since it is the parameter along which we observe the most bias. Although we use the same network architecture and fix all other settings, the detection performance is significantly reduced. We detected signs of overfitting during training, likely because many of the training hyperparameters were tuned for the original setting and higher variance in the noise and signal classes.

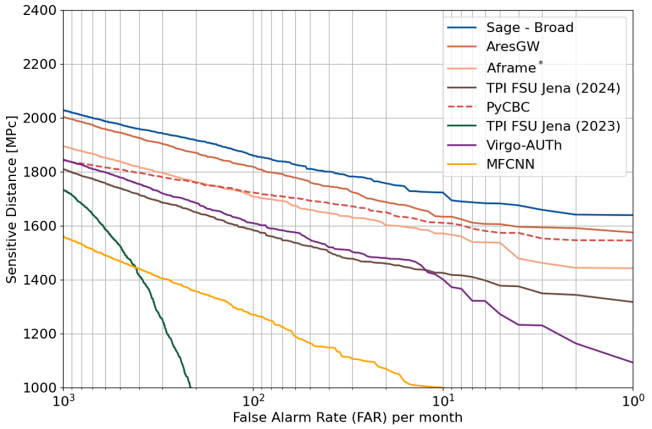


FIG. 19. The sensitive distance metric as a function of FAR per month for Sage—Broad, other ML pipelines, and PyCBC on the D4 testing dataset. The AresGW results were obtained from [53], Aframe and TPI FSU Jena (2024) results via private communication, and all other results were obtained from [21]. The Aframe results are marked with an asterisk (*) as their pipeline is not optimized for the D4 mass distribution in MLGWSC-1 [21].

We fine-tune the hyperparameters of the network to work on the reduced dataset for a fairer comparison. Although the Sage architecture reduces some of the biases, the distribution of the parameters, the amount of variation, and the training strategy are paramount to achieving a good detection performance. Figure 20(b) shows an increase in performance due to the fine-tuning but does not compare well with our best results. All ablation studies henceforth have been fine-tuned to account for the corresponding changes made.

Study II—Using a simpler network architecture. In this study, we use a 1D ResNet-50 [100], which is comparable in size to the networks used in [53,54,56]. Because of the lack of relevant inductive biases (a network architecture or training strategy particularly designed for the problem) in the standard 1D ResNet, we observe in Fig. 20(c) that the detection sensitivity has significantly degraded compared to Sage. This study indicates that the architectures used in AresGW [53], enhanced-AresGW [54], and Aframe [56] would not experience a significant increase in detection sensitivity even if they were trained on 160 days of real O3 noise, 80 million training samples of length 12 s, and the same training strategies as Sage. This empirically justifies the requirement of appropriate inductive biases.

Study III—Using a simple (but deeper) network architecture. For the third study, we use a 1D ResNet-152 [100]. We chose this architecture for two reasons: (i) This is

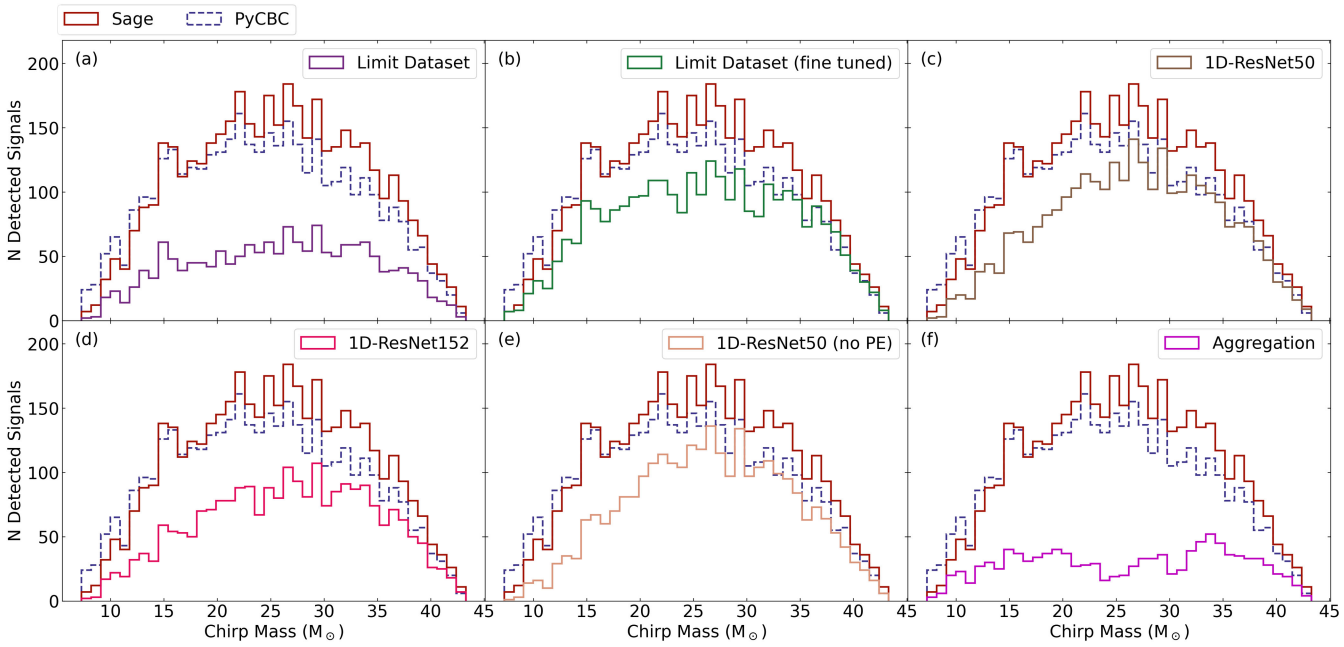


FIG. 20. 1D chirp mass histograms of detected signals for all ablation studies compared with Sage—Broad and PyCBC at an FAR of 1 per month. Panels (a) and (b) correspond to ablation study I (decreased noise and signal realizations), panel (c) shows study II (using a simpler network architecture), panel (d) shows study III [using a simple (but deeper) network architecture], panel (e) shows study IV (abandon regularization), and panel (f) shows study V (effects of aggregation).

a fairer comparison to Sage, as the 1D ResNet-50 is significantly smaller in terms of the number of trainable parameters, and (ii) deeper networks are known to learn long-term dependencies better [72], which also offers an advantage over the 1D ResNet-50. However, we see in Fig. 20(d) that the 1D ResNet-152 performs worse than the much shallower 1D ResNet-50. This is because we fixed the number of training iterations to be the same for all ablation studies. Bigger networks are typically less data efficient than shallower networks [130], which explains the slightly worse performance in comparison to a 1D ResNet-50. However, this is true *only* when the proper inductive biases are not used. The 1D ResNet-152 and Sage have nearly the same number of trainable parameters, with the former being slightly larger. The reduction in performance seen in Fig. 20(d) is, thus, purely due to the lack of proper inductive bias. The Sage architecture allows for significantly higher data efficiency.

Study IV—Abandon regularization. All previous ablation studies have predicted a point estimate of the chirp mass and time of coalescence of the supposed GW signal in each sample. Section II G of the methodology describes this point estimate as a regularization method that provides an additional inductive bias to distinguish between the signal and noise classes. We repeat study II but remove this regularization and see in Fig. 20(e) that it leads to a comparative reduction in detection performance with Fig. 20(c). Since the training seeds were exactly the same between studies II and IV, we can safely conclude that the difference is purely due to the lack of this regularization feature.

Study V—Effects of aggregation. For the final study, we emphasize the aggregating effects of supervised learning by using an architecture that is not well suited to handling glitches in the data. Instead of our CBAM ResNet-50 back-end classifier, we use a Res2Net [131], a multiscale backbone to the standard ResNet architecture. We see that, in Fig. 20(f), this relatively minor change led to a drastic reduction in detection performance. We found that certain architectures led to a completely invariant ranking statistic for all signals above a certain threshold in SNR (let us name this T_{snr}). This threshold was typically present close to the signal-noise class decision boundary. Say we have 1 month of unseen testing data. The presence of a single unseen difficult-to-classify glitch in this dataset could generate a trigger with a ranking statistic close to T_{snr} and instantly reduce the detection sensitivity at an FAR of 1 per month. This sort of *aggregation* of triggers close to the decision boundary can be mildly alleviated when using larger networks and significantly mitigated when using networks with proper inductive biases.

IV. RUNNING SAGE ON LIVE DATA

Sections III A 1, III A 3, and III A 4 prove that Sage is able to achieve highly sensitive detection performance with good glitch rejection capability and limited extrapolation capability with noise PSDs. Thus, we believe that Sage can handle real-time detection of GW alongside other existing production-level pipelines. We dedicate this section to detail minor modifications to Sage that will allow for its usage on an unknown live data feed.

Say we want to run Sage on the fifth observing run (O5). Before running a search on live data, we must first train the model with noise realizations that resemble the new data, to bridge the gap due to different noise PSDs. We can obtain a small amount of noise PSDs from the new data and recolor previously known noise realizations from O3 and O4, giving us a large dataset. These data can then be used to fine-tune Sage over one or two epochs, which allows it to adapt to the new noise PSDs. To achieve a low enough FAR, we can modify Sage in one of the following ways.

Method 1. Our runs in the injection study assume a known background dataset to estimate the FAR of an event. In the case of live data, we can retain the current multidetector architecture of Sage and perform a time-slide analysis [91] of the strain data with limited data to achieve low FARs. To perform a time-sliding technique, the model must first be trained on a dataset that includes a signal in one detector but not the other. This will incentivize the model to produce appropriate null triggers when operating on time-slid data. However, this will use unnecessary learning capacity (since time-slid data are not of interest astrophysically) and will likely increase training times. For each time-slid data sample, the model must run in evaluation mode; to achieve an FAR of 1 per year, we must evaluate the model on 1 year of time-slid data. The primary disadvantage of this method is that going to very low FARs might become infeasible given evaluation times.

Method 2. We can run Sage on a single-detector coincident analysis setting and perform time sliding. Background estimation can be performed similar to PyCBC Live, by storing a rolling buffer of detector triggers and performing the time-sliding technique to estimate the FAR. Shifting the triggers beyond the light travel time between detectors will negate some background triggers via nonphysical coincidences. Contrary to the first method, this will lead to highly efficient time sliding of *triggers* instead of the strain itself. However, a single-detector ML model will not be able to take advantage of some of the architectural features introduced in Sec. II G, like channel-wise attention in the back-end classifier. This might reduce detection efficiency.

Running Sage on O3. Say we want to conduct an offline search on O3 data with Sage. Since Sage is trained on the entirety of O3 noise data (as labeled by current matched-filtering searches), there is the natural concern that our training dataset possibly includes as-of-yet-undetected signals in the noise class. However, given that the ratio between the volume of undetected signals to genuine noise is negligible, the following biases will prevent the model from learning undetected signals as noise: (i) the *bias due to limited sample representation* and (ii) the significant imbalance between noise and signal features in the noise class (*bias due to limited feature representation*).

V. LIMITATIONS OF SAGE

Persistent biases. Although we tried to address the biases mentioned in Sec. I, our best results in Fig. 13 are still marginally biased against lower chirp masses. An attempt to alleviate all mentioned biases, as seen for the Sage—Annealed train run in Fig. 10, leads to an ≈13.3% decrease in the number of signals detected at an FAR of 1 per month when compared to PyCBC.

Evaluation speed. A larger testing dataset would allow for performance evaluations at lower FARs. However, even at the high evaluation speed of the analysis, the current coherent strategy is limited due to the issues raised in Sec. IV, method 1. Since evaluating on 1 month of testing data takes ≈16 h, achieving an FAR of 1 in 10 years, for example, would require ≈80 days. Even if a model had an evaluation speed of 30 min for 1 month of testing data, it would still take 2.5 days. Thus, Sec. IV method 1 becomes quickly impractical.

Mild train-test overlap. The best result we present uses noise PSDs obtained from the testing dataset to recolor the training noise realizations. The resulting recolored noise realization is technically new, since the testing data PSD is only an estimate and whitened noise realization (before being recolored) is different from testing data. Although the network does not exhibit any signs of overfitting, this method still introduces mild train-test overlap. It was shown in Sec. III A 4 that the performance gain was *not* due to this overlap, but the result of a broader distribution of noise PSDs in the training dataset. We can, thus, alleviate the mild train-test overlap by generating artificial PSDs using a normalizing flow [132] and retraining Sage once before an online search with some representative PSDs. It is possible to retrain an ML model such that it does not need further retraining throughout an observing run, as evidenced by [133].

Small testing dataset. The number of glitches, although substantial [$\mathcal{O}(10^4)$] within the month of O3a testing noise, is not sufficient to fully analyze the glitch rejection capability of Sage. Moreover, there is no guarantee that

the glitch profiles of *all* glitch types were fully learned given the imbalance of data for different glitch types (see Fig. 23 in Appendix C).

Need for large training dataset. Our current work requires a large training dataset in order to reduce biases. Providing sufficient variation to all GW parameters and including all known noise realizations will prove difficult in the future. Thus, we have to work toward creating an information-efficient pipeline that is able to adapt to unforeseen changes in noise characteristics without the need for retraining.

VI. CONCLUSION

We introduce Sage, an ML-based BBH detection pipeline, that is shown to be capable of detecting ≈11.2% and ≈48.29% more signals than the benchmark PyCBC submission in [21] and the AresGW results in [53] for the MLGWSC-1 injection study, respectively, at an FAR of one per month in O3a noise. We achieve this performance by identifying a nonexhaustive set of 11 interconnected biases that are present in the usage of ML for the GW detection problem and creating mitigation tactics and training strategies to address them concurrently. We prove that Sage is robust to OOD variation in noise power spectral densities and can reject non-Gaussian transient noise artifacts effectively. Explanations and rationale have been provided for all method choices, and an ablation study was conducted to provide evidence for our claims. Our work addresses the need for interpretability in the application of ML to GW detection.

The biases, mitigation tactics, and training strategies that we provide are general, and the salient ideas are transferable to the detection of other sources of GWs and to other domains of GW data analysis. We reiterate some of our primary empirical contributions for the application of ML to the GW detection problem.

Negating the need for PSD estimation. Contrary to previous ML-based detection pipelines operating on real detector noise [53,54,56,64], we show that Sage does not need to use a PSD estimate for whitening any input sample (see Sec. II). We use a single approximate PSD instead for all samples.

Mitigating biases might aid in glitch rejection. In Sec. III A 3, we obtain all the triggers generated by Sage and PyCBC during glitch events in one month of O3a noise and prove that Sage is capable of effectively rejecting glitches. We also argue about the potential benefits of mitigating biases for glitch rejection.

OOD PSDs are good for classification. In Sec. III A 4, we show that using a broader range of noise PSDs for the training data, which could even be OOD to the testing dataset, greatly aids in detection performance.

Real noise is better than Gaussian noise for training. We show in Sec. III A 4 that models trained with real detector noise produce more confident classifiers than models trained with simulated colored Gaussian noise.

Parameter efficiency via inductive biases. Via an ablation study (see Sec. III C), we show that a model that is larger than Sage in terms of number of trainable parameters is not able to achieve nearly the same detection sensitivity as Sage, *if* it does not have the proper inductive biases.

ACKNOWLEDGMENTS

N. N. extends special thanks to Will Benoit and Ondřej Zelenka for running their respective ML-based search pipelines, [56,64], on the D4 testing dataset provided in MLGWSC-1 [21]. We appreciate the useful comments from Thomas Dent and Nikolaos Stergioulas on our paper. N. N. acknowledges and appreciates the support of Joseph Bayley, Michael Williams, and Christian Chapman-Bird. We also extend our sincere gratitude to the Physics and Astronomy-Machine Learning group members from the University of Glasgow, for their fruitful weekly meetings. N. N. is supported by the College Scholarship offered by the School of Physics and Astronomy (2021–2025), University of Glasgow. C. M. is supported by STFC Grant No. ST/Y004256/1. This material is based upon work supported by NSF's LIGO Laboratory, a major facility fully funded by the National Science Foundation.

DATA AVAILABILITY

The data that support the findings of this article are openly available [75].

APPENDIX A: TESTING DATASET PARAMETER DISTRIBUTIONS

Figure 21 shows the gravitational-wave parameter distributions for the D4 testing dataset introduced in the MLGWSC-1 [21]. This was generated using the data-generation ini file provided in the MLGWSC-1 GitHub repository for use with PyCBC (see [134]).

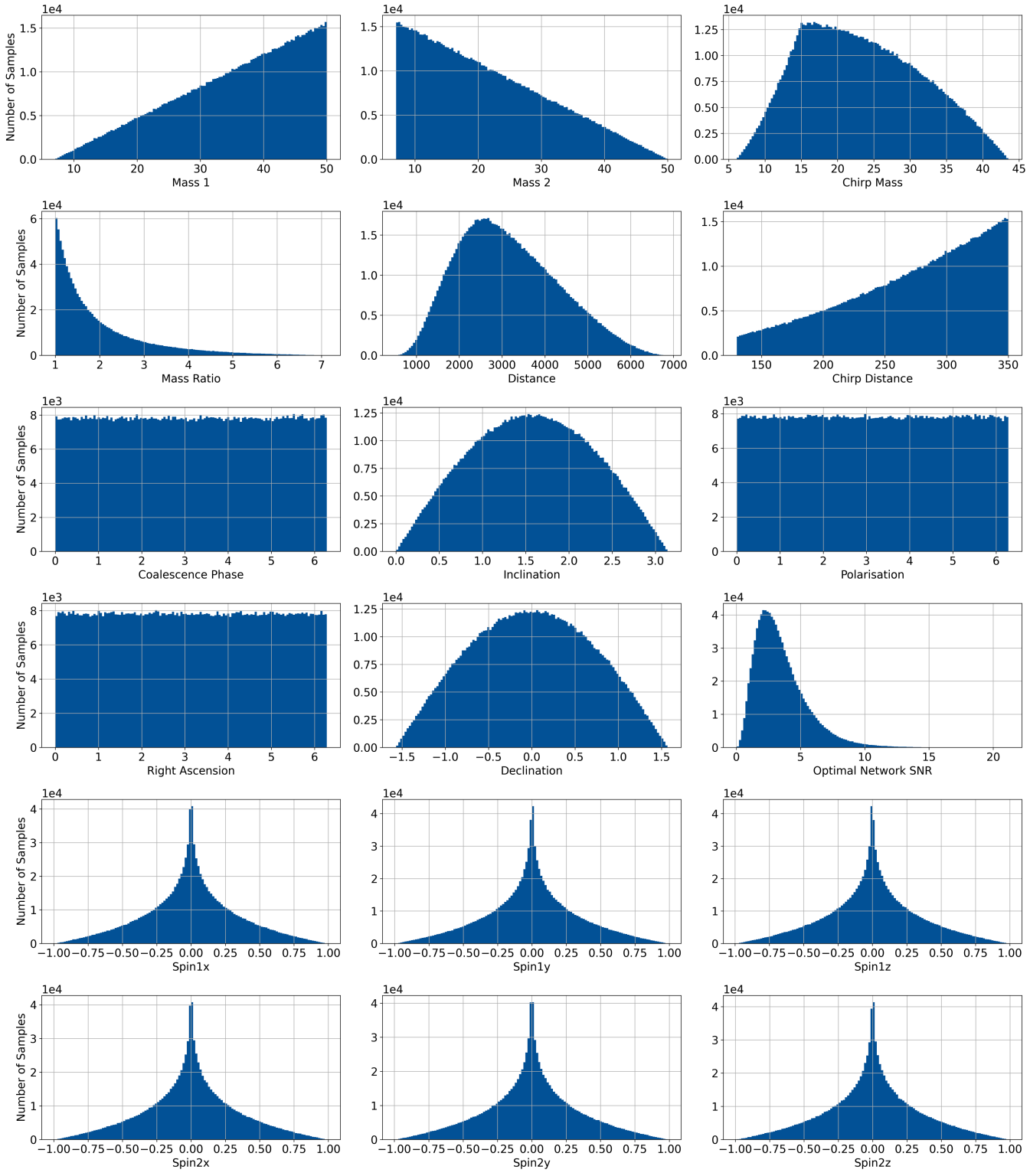


FIG. 21. 1D histograms of GW parameters showing the parameter distributions based on the D4 testing dataset generated from 10^6 samples. The distributions are fully defined in Table I.

APPENDIX B: SAGE—BROAD DETECTION SENSITIVITY

Figure 22 shows the 1D histograms of detected events along different gravitational-wave parameters at an FAR of 1 per month for Sage—Broad. Figures 13, 17, and 18 show the same for a limited number of parameters—chirp mass, mass ratio, spin-1z, spin-2z, and optimal network SNR. Here, we provide a full comparison of Sage—Broad and PyCBC.

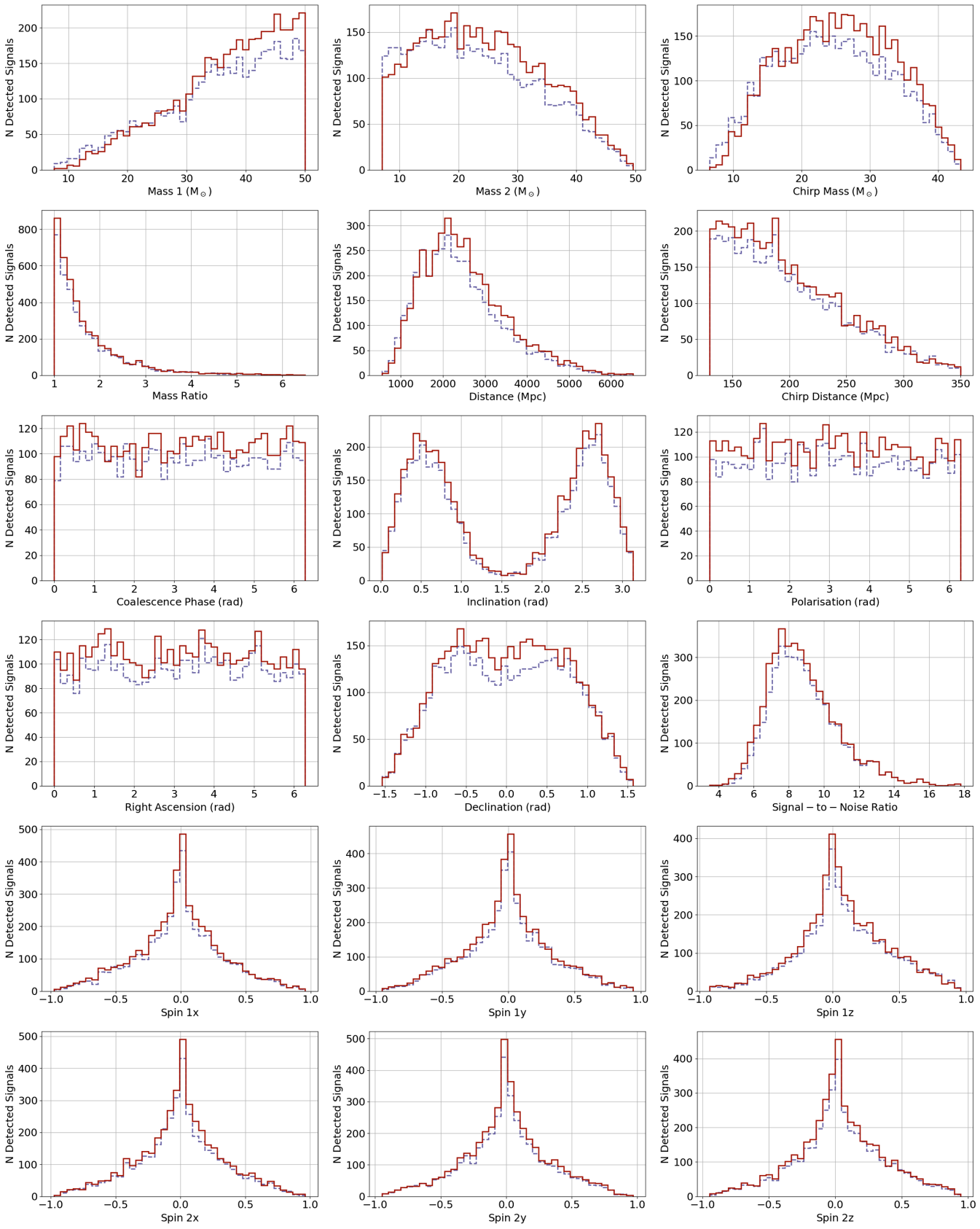


FIG. 22. 1D chirp mass histograms of different GW parameters showing the number of detections made by Sage—Broad and PyCBC at an FAR of 1 per month for the D4 testing dataset.

APPENDIX C: NETWORK BEHAVIOR FOR DIFFERENT GLITCH TYPES

We investigated a little further into the comparison of generated triggers for different types of glitches (see Fig. 23). We arrived at three major observations.

Observation 1. The training component mass distribution and its related learning biases determine the glitch types that the ML model finds most difficult to classify.

Observation 2. The Sage—Annealed train run typically produced fewer glitch triggers between an FAR of 10^4 per month to 1 per month, when compared to Sage—Broad.

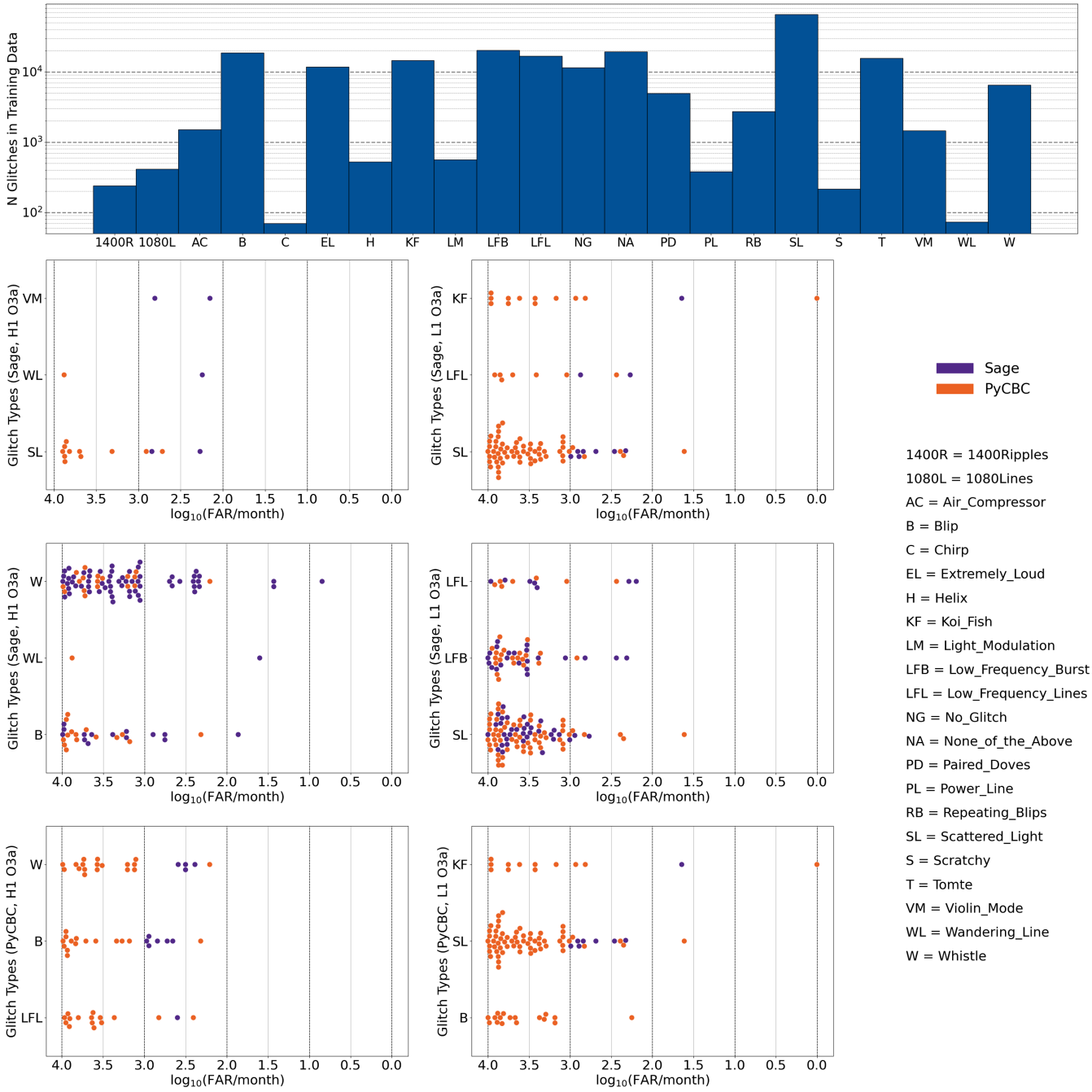


FIG. 23. Glitch triggers from different glitch types at FARs from 10^4 per month to 1 per month for different Sage runs and PyCBC. The left and right columns correspond to the H1 and L1 detectors, respectively. The top row corresponds to the Sage—Annealed train run, the middle row to the Sage—Broad run, and the bottom row to PyCBC. The glitch types chosen for each panel produced the top three triggers for that configuration ("top" here refers to triggers with the lowest observed FARs).

Observation 3. We did not observe any noticeable correlation between the difficult-to-classify glitch types and their number of occurrences in the training data. In particular, glitch types that were not significantly present in the training dataset were not any more difficult to classify than the glitch types that were present in abundance. This might be an artifact of a limited testing dataset.

APPENDIX D: PARAMETER POINT ESTIMATE

Figure 24 shows parity plots of true vs predicted chirp mass and coalescence time for the validation dataset from Sage—Annealed train and Sage—Broad, evaluated at the epoch with the lowest validation loss. These point estimates are included primarily to provide an auxiliary inductive bias during training while also serving to passively diagnose parameter-specific training biases.

Diagnosing biases. For both Sage—Annealed train (Fig. 24, top) and Sage—Broad (Fig. 24, bottom), the chirp mass is typically underestimated near the upper end of its range, even at high network SNR. Two possible causes are (i) insufficient representation of high chirp mass signals in the training data and (ii) model limitations in capturing short-duration waveform morphology. The second is unlikely, as it would produce random errors rather than systematic bias. Supporting the first explanation, Sage—Broad, which trains on $U(m_1, m_2)$, shows reduced bias compared to Sage—Annealed train, which trains on $U(\tau_0, \tau_3)$ and is more underrepresented at high chirp masses.

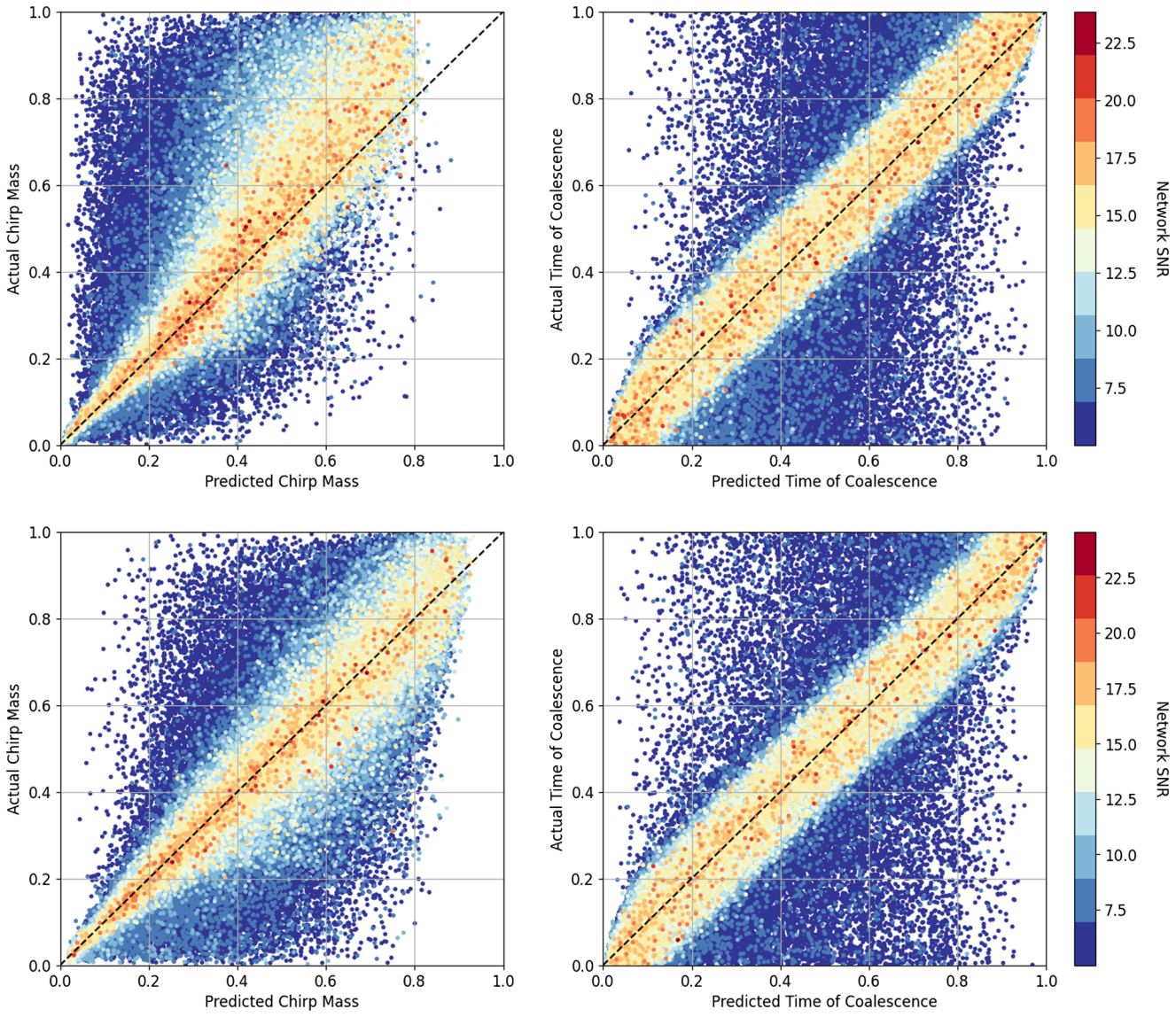


FIG. 24. Actual value of GW parameters: chirp mass $\mathcal{M}_c$ and time of coalescence t_c, plotted against the network predicted value for a validation epoch in different Sage runs. The points are colored based on the network optimal SNR of their corresponding gravitational-wave signal. The top row corresponds to the Sage—Annealed train run and the bottom row to the Sage—Broad run.

[1] F. Chollet, arXiv:1911.01547.
[2] Z. Fang, Y. Li, J. Lu, J. Dong, B. Han, and F. Liu, arXiv:2210.14707.
[3] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, ACM Comput. Surv., 10.1145/3457607 (2021).
[4] I. H. Sarker, *Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions* (Springer-Verlag, Berlin, Heidelberg, 2021).
[5] S. V. Dhurandhar and B. S. Sathyaprakash, Phys. Rev. D **49**, 1707 (1994).
[6] B. J. Owen and B. S. Sathyaprakash, Phys. Rev. D **60**, 022002 (1999).
[7] S. Babak *et al.*, Phys. Rev. D **87**, 024033 (2013).
[8] S. A. Usman *et al.*, Classical Quantum Gravity **33**, 215004 (2016).
[9] A. D. Whalen, IEEE Commun. Mag. **22**, 37 (1984).
[10] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Classical Quantum Gravity **37**, 055002 (2020).
[11] R. Abbott *et al.* (LIGO Scientific Collaboration, Virgo Collaboration, and KAGRA Collaboration), Phys. Rev. X **13**, 041039 (2023).
[12] T. Accadia *et al.*, in *Proceedings of the 16th Lomonosov Conference on Elementary Particle Physics* (2015), pp. 261–270.
[13] B. P. Abbott *et al.*, Living Rev. Relativity **21**, 3 (2018).
[14] LIGO Scientific Collaboration *et al.*, Classical Quantum Gravity **32**, 074001 (2015).
[15] A. Buikema *et al.*, Phys. Rev. D **102**, 062003 (2020).
[16] M. Tse *et al.*, Phys. Rev. Lett. **123**, 231107 (2019).
[17] D. V. Martynov *et al.*, Phys. Rev. D **93**, 112004 (2016).
[18] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Phys. Rev. Lett. **116**, 131103 (2016).
[19] G. M. Harry (for LIGO Scientific Collaboration), Classical Quantum Gravity **27**, 084006 (2010).
[20] F. Acernese *et al.* (Virgo Collaboration), Phys. Rev. Lett. **123**, 231108 (2019).
[21] M. B. Schäfer *et al.*, Phys. Rev. D **107**, 023021 (2023).
[22] B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown, and J. D. E. Creighton, Phys. Rev. D **85**, 122006 (2012).
[23] A. H. Nitz, T. Dent, T. D. Canton, S. Fairhurst, and D. A. Brown, Astrophys. J. **849**, 118 (2017).
[24] A. H. Nitz, T. Dal Canton, D. Davis, and S. Reyes, Phys. Rev. D **98**, 024050 (2018).
[25] C. Rover, J. Phys. Conf. Ser. **228**, 012008 (2010).
[26] A. C. Searle, in *Proceedings of the 12th Gravitational Wave Data Analysis Workshop* (2008), arXiv:0804.1161.
[27] P. Kumar and T. Dent, Phys. Rev. D **110**, 043036 (2024).
[28] T. Dent and J. Veitch, Phys. Rev. D **89**, 062002 (2014).
[29] J. Yan, M. Avagyan, R. E. Colgan, D. Veske, I. Bartos, J. Wright, Z. Márka, and S. Márka, Phys. Rev. D **105**, 043006 (2022).
[30] S. Babak, R. Balasubramanian, D. Churches, T. Cokelaer, and B. S. Sathyaprakash, Classical Quantum Gravity **23**, 5477 (2006).
[31] J. R. Gair, L. Barack, T. Creighton, C. Cutler, S. L. Larson, E. S. Phinney, and M. Vallisneri, Classical Quantum Gravity **21**, S1595 (2004).
[32] D. C. Srivastava and S. K. Sahay, Mon. Not. R. Astron. Soc. **337**, 322 (2002).
[33] J. Aasi *et al.* (LIGO Scientific Collaboration), Classical Quantum Gravity **32**, 074001 (2015).
[34] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Classical Quantum Gravity **33**, 134001 (2016).
[35] M. Cabero *et al.*, Classical Quantum Gravity **36**, 15 (2019).
[36] S. Biscoveanu, C.-J. Haster, S. Vitale, and J. Davies, Phys. Rev. D **102**, 023008 (2020).
[37] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Classical Quantum Gravity **35**, 065010 (2018).
[38] C. Messick *et al.*, Phys. Rev. D **95**, 042001 (2017).
[39] B. Allen, Phys. Rev. D **71**, 062001 (2005).
[40] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. X **9**, 031040 (2019).
[41] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. D **93**, 122003 (2016).
[42] A. H. Nitz, Classical Quantum Gravity **35**, 035016 (2018).
[43] R. Balestriero, A. I. Humayun, and R. Baraniuk, arXiv:2408.04809.
[44] H. Ye, G. Y. Li, and B.-H. Juang, IEEE Wireless Commun. Lett. **7**, 114 (2018).
[45] J. Chen, Y. Wang, and D. Wang, Language Process. **22**, 1993 (2014).
[46] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, arXiv:2205.01917.
[47] K. Kawaguchi, L. Pack Kaelbling, and Y. Bengio, arXiv:1710.05468.
[48] G. V. Pérez, C. Q. Camargo, and A. A. Louis, arXiv:1805.08522.
[49] M.-P. Jarabo-Amores, M. Rosa-Zurera, R. Gil-Pita, and F. Lopez-Ferreras, IEEE Trans. Signal Process. **57**, 4175 (2009).
[50] M.-P. Jarabo-Amores, D. d. la Mata-Moya, R. Gil-Pita, and M. Rosa-Zurera, EURASIP J. Appl. Signal Process. **2013**, 44 (2013).
[51] D. George and E. A. Huerta, Phys. Lett. B **778**, 64 (2018).
[52] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).
[53] P. Nousi, A. E. Koloniari, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, Phys. Rev. D **108**, 024022 (2023).
[54] A. E. Koloniari, E. C. Koursoumpa, P. Nousi, P. Lampropoulos, N. Passalis, A. Tefas, and N. Stergioulas, Mach. Learn. Sci. Tech. **6**, 015054 (2025).
[55] M. B. Schäfer and A. H. Nitz, Phys. Rev. D **105**, 043003 (2022).
[56] E. Marx, W. Benoit, A. Gunny, R. Omer, D. Chatterjee, R. C. Venterea, L. Wills, M. Saleem, E. Moreno, R. Raikman, E. Govorkova, D. Rankin, M. W. Coughlin, P. Harris, and E. Katsavounidis, arXiv:2403.18661.
[57] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Phys. Rev. D **100**, 063015 (2019).
[58] C. Cutler and É. E. Flanagan, Phys. Rev. D **49**, 2658 (1994).
[59] C. Mingard, J. Skalse, G. V. Pérez, D. Martínez-Rubio, V. Mikulik, and A. A. Louis, arXiv:1909.11522.
[60] G. V. Pérez, C. Q. Camargo, and A. A. Louis, arXiv:1805.08522.
[61] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, arXiv:1810.00825.
[62] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, in *Proceedings of the 26th International Conference on*

Neural Information Processing Systems—Volume 1, NIPS'13 (Curran Associates Inc., Red Hook, NY, 2013), pp. 1196–1204.
[63] M. Denil and T. Trappenberg, in *Proceedings of the 23rd Canadian Conference on Advances in Artificial Intelligence, AI'10* (Springer-Verlag, Berlin, Heidelberg, 2010), pp. 220–231.
[64] O. Zelenka, B. Brügmann, and F. Ohme, arXiv:2402.07492.
[65] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning* (Springer, New York, 2006), Vol. 4.
[66] M. S. Masiha, A. Gohari, M. H. Yassaee, and M. R. Aref, in *Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT)* (IEEE, New York, 2021), pp. 2912–2917.
[67] R. J. N. Baldock, H. Maennel, and B. Neyshabur, arXiv:2106.09647.
[68] Y. Bengio, P. Simard, and P. Frasconi, IEEE Trans. Neural Networks **5**, 157 (1994).
[69] S. E. Hihi and Y. Bengio, in *Neural Information Processing Systems* (1995), https://api.semanticscholar.org/CorpusID:2843869.
[70] S. Hochreiter and J. Schmidhuber, Neural Comput. **9**, 1735 (1997).
[71] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, in *Neural Information Processing Systems* (2017), https://api.semanticscholar.org/CorpusID:13756489.
[72] W. Luo, Y. Li, R. Urtasun, and R. S. Zemel, arXiv:1701.04128.
[73] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019), pp. 5301–5310.
[74] R. Basri, M. Galun, A. Geifman, D. W. Jacobs, Y. Kasten, and S. Kritchman, in *Proceedings of the International Conference on Machine Learning* (2020).
[75] N. Nagarajan, nnarenraju/sage: First DOI version of Sage, 10.5281/zenodo.17290134.
[76] G. Pratten, C. García-Quirós, M. Colleoni, A. Ramos-Buades, H. Estellés, M. Mateu-Lucena, R. Jaume, M. Haney, D. Keitel, J. E. Thompson, and S. Husa, Phys. Rev. D **103**, 104056 (2021).
[77] C. Mills and S. Fairhurst, Phys. Rev. D **103**, 024042 (2021).
[78] R. Green, C. Hoy, S. Fairhurst, M. Hannam, F. Pannarale, and C. Thomas, Phys. Rev. D **103**, 124023 (2021).
[79] R. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. X **11**, 021053 (2021).
[80] https://gwosc.org/datadef/.
[81] J. Solomon and O. M., PSD computations using Welch's method (1991), https://www.osti.gov/biblio/5688766.
[82] J. Glanzer *et al.*, Gravity spy machine learning classifications of LIGO glitches from observing runs O1, O2, O3A, and O3B (2021), 10.5281/zenodo.5649212.
[83] https://github.com/gwastro/ml-mock-data-challenge-1.
[84] S. Babak, R. Balasubramanian, D. Churches, T. Cokelaer, and B. S. Sathyaprakash, Classical Quantum Gravity **23**, 5477 (2006).
[85] B. S. Sathyaprakash and S. V. Dhurandhar, Phys. Rev. D **44**, 3819 (1991).
[86] T. Cokelaer, Phys. Rev. D **76**, 102004 (2007).
[87] https://gwosc.org/eventapi/html/allevents/.
[88] https://gwosc.org/timeline/show/O3b_4KHZ_R1/H1_DATA*L1_DATA*V1_DATA/1256655618/12708000/.
[89] J. P. Cohen, T. Cao, J. D. Viviano, C.-W. Huang, M. Fralick, M. Ghassemi, M. Mamdani, R. Greiner, and Y. Bengio, Cmaj **193**, E1391 (2021).
[90] E. Beede, E. Baylor, F. Hersch, A. Iurchenko, L. Wilcox, P. Ruamviboonsuk, and L. M. Vardoulakis, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, CHI '20* (Association for Computing Machinery, New York, NY, 2020), pp. 1–12.
[91] M. Wąs, M.-A. Bizouard, V. Brisson, F. Cavalier, M. Davier, P. Hello, N. Leroy, F. Robinet, and M. Vavoulidis, Classical Quantum Gravity **27**, 015005 (2010).
[92] M. B. Schäfer, F. Ohme, and A. H. Nitz, Phys. Rev. D **102**, 063015 (2020).
[93] D. P. Kingma and J. Ba, arXiv:1412.6980.
[94] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 2014), p. 580.
[95] K. He, X. Zhang, S. Ren, and J. Sun, IEEE Trans. Pattern Anal. Mach. Intell. **37**, 1904 (2015).
[96] K. He, G. Gkioxari, P. Dollár, and R. Girshick, in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)* (IEEE, New York, 2017), pp. 2980–2988.
[97] C. Messenger, C. Zerafa, E. Cuoco, M. J. Williams, and W. Reade, G2net gravitational wave detection (2021), https://www.kaggle.com/competitions/g2net-gravitational-wave-detection.
[98] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2017).
[99] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (IEEE, New York, 2019), p. 3701.
[100] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2016), pp. 770–778.
[101] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, in *Computer Vision—ECCV 2018: 15th European Conference, Munich, Germany, 2018, Proceedings, Part VII* (Springer-Verlag, Berlin, Heidelberg, 2018), pp. 3–19.
[102] D. Bank, N. Koenigstein, and R. Giryes, Deep Learning in Science (2021), https://api.semanticscholar.org/CorpusID:212717965.
[103] J. Hu, L. Shen, and G. Sun, in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 2018), pp. 7132–7141.
[104] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2016), pp. 2921–2929.
[105] V. Tiwari and S. Fairhurst, Astrophys. J. Lett. **913**, L19 (2021).

[106] Z. Cui, W. Chen, and Y. Chen, arXiv:1603.06995.
[107] X. Liu, B. Xu, S. Cao, and L. Zhang, J. Comput. Phys. **506**, 112944 (2024).
[108] X. Chai, W. Cao, J. Li, H. Long, and X. Sun, IEEE Trans. Geosci. Remote Sens. **62**, 1 (2024).
[109] R. Fang and Y. Xu, arXiv:2410.16105.
[110] B. L. Edelman, S. Goel, S. M. Kakade, and C. Zhang, arXiv:2110.10090.
[111] S. Hao, D.-H. Lee, and D. Zhao, Trans. Res. Part C **107**, 287 (2019).
[112] https://github.com/gwastro/ml-mock-data-challenge-1/blob/master/evaluate.py.
[113] A. H. Nitz, S. Kumar, Y.-F. Wang, S. Kastha, S. Wu, M. Schäfer, R. Dhurkunde, and C. D. Capano, Astrophys. J. **946**, 59 (2023).
[114] D. Davis and M. Walker, Galaxies **10**, 12 (2022).
[115] M. Cabero *et al.*, Classical Quantum Gravity **36**, 15 (2019).
[116] L. Nuttall *et al.*, Classical Quantum Gravity **32**, 245005 (2015).
[117] A. H. Nitz, T. Dent, T. Dal Canton, S. Fairhurst, and D. A. Brown, Astrophys. J. **849**, 118 (2017).
[118] M. Cabero *et al.*, Classical Quantum Gravity **36**, 15 (2019).
[119] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, Proc. Natl. Acad. Sci. U.S.A. **114**, 3521 (2017).
[120] M. Toneva, A. Sordoni, R. T. des Combes, A. Trischler, Y. Bengio, and G. J. Gordon, arXiv:1812.05159.
[121] M. P. Kumar, B. Packer, and D. Koller, in Proceedings of the 23rd International Conference on Neural Information Processing Systems—Volume 1, NIPS'10 (Curran Associates Inc., Red Hook, NY, 2010), pp. 1189–1197.
[122] F. Robinet, N. Arnaud, N. Leroy, A. Lundgren, D. Macleod, and J. McIver, SoftwareX **12**, 100620 (2020).
[123] S. Sahayam, J. Zakkam, and U. Jayaraman, arXiv:2304.03486.
[124] Y. Suh, B. Han, W. Kim, and K. M. Lee, in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2019), pp. 7244–7252.
[125] R. Yu, Z. Dou, S. Bai, Z. Zhang, Y. Xu, and X. Bai, in *Computer Vision—ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI* (Springer-Verlag, Berlin, Heidelberg, 2018), pp. 196–212.
[126] D. Arpit, S. Jastrzundefinedbski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, in *Proceedings of the 34th International Conference on Machine Learning—Volume 70, ICML'17* (JMLR.org, 2017), pp. 233–242.
[127] S. Chatterjee, in *Proceedings of the International Conference on Machine Learning* (PMLR, 2018), pp. 755–763.
[128] X. Wang, Y. Chen, and W. Zhu, IEEE Trans. Pattern Anal. Mach. Intell. **44**, 4555 (2022).
[129] R. Meyes, M. Lu, C. W. de Puiseau, and T. Meisen, arXiv:1901.08644.
[130] Y. Dauphin and Y. Bengio, arXiv:1301.3583.
[131] S. Gao, M.-M. Cheng, K. Zhao, X. Zhang, M.-H. Yang, and P. H. S. Torr, IEEE Trans. Pattern Anal. Mach. Intell. **43**, 652 (2019).
[132] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, J. Mach. Learn. Res. **22**, 2617 (2021).
[133] J. Wildberger, M. Dax, S. R. Green, J. Gair, M. Pürrer, J. H. Macke, A. Buonanno, and B. Schölkopf, Phys. Rev. D **107**, 084046 (2023).
[134] https://github.com/gwastro/ml-mock-data-challenge-1/blob/master/ds4.ini.