Machine-learning pipeline for real-time detection of gravitational waves from compact binary coalescences

Ethan Marx,[1,2] William Benoit,[3] Alec Gunny,[1,2] Rafia Omer,[3] Deep Chatterjee,[1,2] Ricco C. Venterea,[3,4] Lauren Wills,[3] Muhammed Saleem,[5,3] Eric Moreno,[1,2] Ryan Raikman,[1,6] Ekaterina Govorkova,[1,2] Malina Desai,[1,2] Jeffrey Krupa,[1] Dylan Rankin,[7] Michael W. Coughlin,[3] Philip Harris,[1] and Erik Katsavounidis[1,2]

[1]*Department of Physics, MIT, Cambridge, Massachusetts 02139, USA*
[2]*LIGO Laboratory, MIT, 185 Albany Street, Cambridge, Massachusetts 02139, USA*
[3]*School of Physics and Astronomy, University of Minnesota, Minneapolis, Minnesota 55455, USA*
[4]*Department of Astronomy, Cornell University, Ithaca, New York 14853, USA*
[5]*Physics Department, University of Texas at Austin, Austin, Texas 78712, USA*
[6]*Department of Physics, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA*
[7]*Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA*

(Received 11 December 2024; accepted 17 January 2025; published 24 February 2025)

The promise of multimessenger astronomy relies on the rapid detection of gravitational waves at very low latencies [$\mathcal{O}(1\ \mathrm{s})$] in order to maximize the amount of time available for follow-up observations. In recent years, neural networks have demonstrated robust nonlinear modeling capabilities and millisecond-scale inference at a comparatively small computational footprint, making them an attractive family of algorithms in this context. However, integration of these algorithms into the gravitational-wave astrophysics research ecosystem has proven nontrivial. Here, we present a machine-learning-based pipeline for the detection of gravitational waves from compact binary coalescences designed to run in low latency. We demonstrate this pipeline to have a fraction of the latency of traditional matched filtering search pipelines while achieving state-of-the-art sensitivity to higher-mass stellar binary black holes.

DOI: 10.1103/PhysRevD.111.042010

I. INTRODUCTION

Gravitational-wave astronomy has developed rapidly since the first direct detection of gravitational waves from a binary black hole merger in 2015 [1], with new detections now a common occurrence [2]. With the fourth observing run (O4) of the LIGO-Virgo-KAGRA (LVK) Collaboration [3–5] already underway, and with future ground- and space-based detectors planned for various points in the next decade [6–8], ever more frequent discoveries of gravitational waves will enable follow-up observation of events across other cosmic messengers such as electromagnetic radiation and astrophysical neutrinos [9–14]. The insights we gain in this era of multimessenger astrophysics will directly correlate with the volume and diversity of data we are able to collect. Already, the first multimessenger event, which included the gravitational-wave event GW170817 [15], the gamma-ray burst GRB170817A [16–19], and the kilonova AT2017gfo [20–22], has led to a gold mine of new science.

While machine learning (ML) is ubiquitous in some areas of physics [23], it has only recently approached a stage of maturity in the gravitational-wave community. To date, there have been a number of machine-learning models proposed for the detection of compact binary coalescences (CBCs)—e.g., [24–28]—but there are none currently running in O4 [29] (though, ML-based unmodeled gravitational-wave searches have seen production usage [30]). This is both a product of well-known infrastructure hurdles separating the development and deployment of machine-learning models [31] as well as a lack of standardized, astrophysically meaningful probes of the sensitivity of these models in the face of nonstationary and transient background noise.

The most well-modeled and frequently observed gravitational-wave events to date are the mergers of binary black hole (BBH) systems [2,32,33]. Their comparatively high number of confirmed detections has given us reasonable models of their population statistics, allowing for astrophysically meaningful measures of search sensitivity. BBH mergers also benefit from a highly localized-in-time signal-to-noise ratio (SNR)[1] profile relative to binary neutron star (BNS) mergers, which are in the sensitive band of the detectors much longer. Studying the ability of neural networks to detect BBH mergers, and in particular what real time use in the LVK detectors looks like in this context,

[1]In the remainder of this paper, SNR refers to the total SNR of a Hanford and Livingston detector network.

represents an important first step toward developing a more thorough understanding of how, and whether, these algorithms can be applied to more challenging signals such as BNSs and what tools and infrastructure would be required to do so.

Here, we present Aframe, a flexible pipeline for detection of BBH mergers using deep learning. The implementation presented here uses a 1D convolutional neural network. Convolutional neural networks have previously been shown to have potential for gravitational-wave detection [34], and we use this architecture, along with aggressive data augmentation techniques, to achieve a sensitivity competitive with matched filtering CBC search pipelines while requiring a significantly lower latency. More broadly, Aframe encompasses a suite of tools for quickly implementing, testing, and deploying new ideas at scale in order to more confidently realize the potential of machine learning in service to gravitational-wave astronomy.

The structure of this paper is as follows. Section II gives a high-level overview of the Aframe algorithm. In Sec. III, we describe the metric we use to measure performance. Section IV describes the datasets used to train and evaluate our network, and the means by which training and evaluation is performed is given in Secs. V and VI, respectively. We discuss the longevity of our model in Sec. VII and the latency and computational requirements in Sec. VIII. Finally, we compare our performance to existing pipelines in Sec. IX and examine a subset of GWTC-3 catalog events in Sec. X.

II. THE AFRAME ALGORITHM

Our neural-network architecture modifies a standard ResNet34 [35], which maps fixed length time series of gravitational-wave strain from two interferometers (here, the Hanford and Livingston LIGO interferometers) to a scalar detection statistic indicating whether a signal is present in the input. The detection statistic is analogous to the SNR output of matched filtering searches; the larger the value, the more likely the neural network believes there is a signal consistent with the training distribution present in the data. Critically, we replace 2D with 1D convolutions to accommodate time-series input. In addition, we replace standard batch normalization (BN) layers [36] with group normalization (GN) layers [37]. While BN layers fit parameters to statistics calculated along the batch dimension, GN layers are fit to statistics calculated from groups of channels. This choice was motivated by differences in the statistical properties of batches during training and inference. During training, there are significantly more signals in each batch than during inference, where most of the batch consists of noise. Thus, during training, BN layers will learn spurious statistical properties that are not present at inference time. GN layers mitigate this problem by learning statistical properties of individual channels. We found that using GN layers improves the agreement between validation and test time metrics, as well as overall testing performance. Good agreement between validation and test metrics is essential for ensuring the best neural network is being selected for deployment. The neural network is trained by minimizing a binary cross entropy loss function with an Adam [38] optimizer. We use a one-cycle learning rate scheduler with cosine annealing [39].

Analyzing data with Aframe involves loading and preprocessing time-series data, breaking it up into short time segments, and then passing these segments through the neural network. The throughput associated with each of these steps can vary drastically, as can the hardware and software necessary to accelerate them. In order to optimize the total throughput of this system, we adopt an inference-as-a-service (IaaS) computing model in which neural-network inference is handled by a dedicated service, to which client applications can send inference requests remotely. Each step in our pipeline is then implemented and scaled independently to most efficiently leverage a fixed pool of heterogeneous computing resources. This model has been shown to be effective in optimizing ML inference in GW astronomy [40], provided that “snapshotting” [41] is used to cache overlapping input data on the server side to avoid redundant data transfer. We adopt this paradigm using an off-the-shelf IaaS implementation, Triton inference server [42], and use the ML inference framework TensorRT to accelerate the neural-network inference step. The ability to scale and distribute a workload is an important part of any search pipeline, and the authors are aware of only one other ML-based CBC detection algorithm that has focused on scalability to arbitrary resources [43]. In the sections below, we compare both our sensitivity and our throughput to this work.

Inference is performed at a rate of 4 Hz (not to be confused with the neural-network throughput; see the discussion of computational requirements below). In other words, we pass windows of data to our neural network for inference such that each window is shifted by 0.25 s. This inference sampling rate reduces the overall compute load without sacrificing search sensitivity (see Sec. VI). These neural-network predictions are then integrated over time using a 1 s top hat filter (see Fig. 1). Because the neural network is trained to encode time translation invariance (see Sec. IV B 2), we expect to see consistently high neural-network responses when analyzing astrophysical signals. Thus, integration provides a mechanism to promote consistently high outputs while rejecting short transients that may correspond to nonastrophysical sources. Finally, the integrated time series of neural-network predictions is clustered to avoid yielding multiple triggers for the same event. The maximum integrated value over an 8 s window is taken as the detection statistic corresponding to a candidate event. The coalescence time can then be estimated by accounting for the integration window length, whitening filter settle-in, and the placement of the

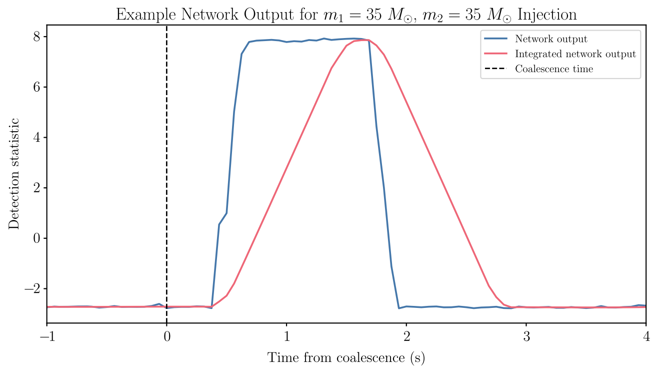

FIG. 1. Example neural-network prediction and integrated neural-network prediction for a $m_1 = 35M_\odot$, $m_2 = 35M_\odot$ signal injection. The coalescence time is plotted as the vertical dashed black line. The 0.75 s gap between coalescence time and the full neural-network activation is due to the whitening filter settle-in time and the placement of injections during training; see Sec. IV B 1.

coalescence time relative to the right edge of the kernel during training (see Fig. 1). The mean absolute time difference between the estimated coalescence time and the injected time for recovered signals in the testing set was 0.123 s with a standard deviation of 0.125 s, which is in line with the inference sampling rate of 4 Hz.

III. SENSITIVE VOLUME

A key metric in understanding a search algorithm's performance is the *sensitive volume*, which is a measure of the region of space in which a pipeline is expected to detect merging binaries. The sensitive volume as a function of the false alarm rate (FAR) is defined by

$$V(\mathcal{F}) = \int \mathrm{d}\mathbf{x}\mathrm{d}\theta \epsilon(\mathcal{F}; \mathbf{x}, \theta)\phi(\mathbf{x}, \theta), \tag{3.1}$$

where ϕ is the distribution of events over spatial coordinates $\mathbf{x}$ and binary system parameters θ and ϵ is the detection efficiency of the pipeline at a false alarm rate $\mathcal{F}$ [44]. Generally, this quantity is estimated using Monte Carlo integration by drawing waveforms from a population model, injecting them into detector strain data, and counting how many produce triggers below a given false alarm rate threshold. If the samples are drawn from within the redshifted volume [45] V_0, with

$$V_0 = \int_{z_{\min}}^{z_{\max}} \mathrm{d}z \frac{\mathrm{d}V_c}{\mathrm{d}z} \frac{1}{1+z}, \tag{3.2}$$

where $\mathrm{d}V_c/\mathrm{d}z$ is the differential comoving volume, then the sensitive volume is approximately

$$V(\mathcal{F}) \approx V_0 \frac{N(\mathcal{F})}{N_{\mathrm{draw}}}, \tag{3.3}$$

where $N(\mathcal{F})$ is the number of signals detected at a FAR less than $\mathcal{F}$ and N_{draw} is the number of injected events.

It is often desired to quantify the sensitivity of an algorithm to different populations. For example, an algorithm's sensitivity may vary with different source masses. Through the technique of importance sampling, it is possible to use one injection set from a broad population to calculate the sensitive volume for several populations. Each injection is weighted by the ratio of the probability of having been drawn from the injected distribution to that of the population distribution of interest [46]:

$$V_{\mathrm{pop}}(\mathcal{F}) \approx \frac{V_0}{N_{\mathrm{inj}}} \sum_{i=1}^{N(\mathcal{F})} \frac{p_{\mathrm{pop}}(\theta_i)}{p_{\mathrm{inj}}(\theta_i)}. \tag{3.4}$$

The Monte Carlo uncertainty on this estimation is [47]

$$(\delta V_{\mathrm{pop}})^2 = \frac{V_0^2}{N_{\mathrm{inj}}^2} \sum_{i=1}^{N(\mathcal{F})} \left(\frac{p_{\mathrm{pop}}(\theta_i)}{p_{\mathrm{inj}}(\theta_i)}\right)^2 - \frac{V_{\mathrm{pop}}^2}{N_{\mathrm{inj}}}. \tag{3.5}$$

The SNR-based rejection performed during the generation of test set waveforms is done to improve this uncertainty. Waveforms that are sampled but have an SNR less than 4 are not injected; however, they are still counted toward N_{draw}. The cut is placed such that any waveforms below the SNR cutoff are not expected to be recovered at any reasonable FAR and so would not contribute to the sensitive volume: whether injected or not, their weight would be zero. This procedure allows us to effectively draw many times more samples than are actually injected, greatly reducing the uncertainty on the sensitive volume. For this analysis, we reweight to the same population distributions used in the sensitive volume analysis conducted in GWTC-3 [2], log-normal distributions about central masses of interest with widths of 0.1. In addition, we enforce time difference of no more than 0.25 s between the recovered and injected coalescence times. This time difference corresponds to the resolution available at an inference sampling rate of 4 Hz. This time resolution can be reduced by increasing the inference sampling rate.

IV. DATA

A. Strain

We train and validate our neural network using open data from the Gravitational Wave Open Science Center [48] between times April 29, 2019 T13:29:25 and May 9, 2019 T13:29:25, corresponding to a ten calendar day period at the beginning of the O3 observing run. The strain data are resampled to 2048 Hz for better computational efficiency. For each interferometer, we query the openly available science mode flag to remove segments with poor data quality. We then select segments for which the science mode flag is active for both the Hanford and Livingston

TABLE I. Priors on parameters used to generate waveforms for both the training and testing sets. The prior is derived from that used in GWTC-3 to assess search pipelines. The component mass distributions are defined in the source frame. "Comoving" refers to uniform in comoving volume.

Parameter	Prior	Limits	Units
Mass of primary	$m_1^{-2.35}$	(5, 100)	$M_\odot$
Mass of secondary	m_2	$(5, m_1)$	$M_\odot$
Redshift	Comoving	(0, 2)	$\cdots$
Polarization angle	Uniform	$(0, \pi)$	rad
Dimensionless spin magnitude	Uniform	(0, 0.998)	$\cdots$
Spin tilt	Sine	$(0, \pi)$	rad
Relative spin azimuthal angle	Uniform	$(0, 2\pi)$	rad
Spin phase angle	Uniform	$(0, 2\pi)$	rad
Orbital phase	Uniform	$(0, 2\pi)$	rad
Right ascension	$(0, 2\pi)$	rad	
Declination	Cosine	$(-\pi/2, \pi/2)$	rad
Inclination angle	Sine	$(0, \pi)$	rad

LIGO interferometers. This amounts to approximately 4.7 days of coincident live time. We reserve the segments that total a minimum of 15 000 s at the end of this period for validating the neural network throughout the training process.

For evaluating the performance reported in Fig. 4, we select data satisfying the above criteria between times May 9, 2019 T13:29:25 and June 8, 2019 T13:29:25, corresponding to a 30 day period immediately after the training period. This amounts to approximately 18 days of coincident live time. During evaluation, time slides (see Sec. IX) of these data are created such that the total desired background live time is achieved. We emphasize that no data used for evaluating the performance of the neural network were used during training or validation. In addition, we train the neural network only with data from before the testing period. This mimics the data availability scenario for real-time application.

B. Waveforms

We use Bilby [49] to simulate 100 000 eight-second-long BBH waveforms at 2048 Hz with the IMRPhenomPv2 approximant [50]. Out of these, 75 000 waveforms are used to train the neural network, and the remaining 25 000 are reserved for validation. To simulate a waveform, a probability distribution is specified on each of the parameters that define a compact binary merger, and random samples are drawn from each. The distribution set used in this work is based on one used for GWTC-3 [51] during O3 to assess the sensitivity of CBC search pipelines and is described in Table I. The sampled parameters are used to compute the time-domain strain for each polarization, h_+ and $h_\times$. The sampled component mass values are defined in the source frame, so conversion to detector frame quantities is performed before generation. The interferometer responses of the intrinsic polarizations are calculated during the training process to allow for real-time data augmentations, as described below in Sec. V.

The same distribution (Table I) is used to simulate signals for the testing dataset. Enough waveforms are generated to fill the background time slides with the waveform coalescence points spaced 24 s apart. As the signals are only 8 s long, they do not overlap. Because waveforms are distributed uniformly in comoving volume, the majority of waveforms are placed at high redshifts, and thus the population is dominated by low SNR signals. So, during the signal generation process, we perform rejection sampling and keep only signals that have an SNR greater than 4. This ensures that we can accurately evaluate sensitivity to an astrophysically motivated population without wasting computation on signals we do not expect to detect [52]. Rejection sampling reduces the uncertainty of a sensitive volume estimate for a fixed amount of analyzed injections (see Sec. III). In total, we generate ~45 000 000 waveforms. Of these, ~3% are used for testing and ~97% are rejected.

We apply several data augmentation techniques during the training process with the goal of providing robust, high entropy data that encodes physics-based knowledge for discriminating signals from noise. Below, we will describe how a training batch is composed, as well as the hyperparameters that control the composition of the batches.

1. Noise sampling

Sampled at 2048 Hz, the entire training dataset is unable to fit onto a single 16 GB V100 GPU at once. Thus, efficient out-of-memory data loading is required to fully utilize the extent of our strain dataset. To do this, we sample

TABLE II. Priors and descriptions of hyperparameters searched over. The best value corresponds to the neural network from the hyperparameter search that produced the highest validation score across all epochs. A neural network trained with these hyperparameters was used to evaluate results reported in Fig. 4. Details on hyperparameters can be found in Sec. V D.

Parameter	Description	Prior	Limits	Best value
lr_{max}	Maximum learning rate	Log uniform	$(10^{-4.5}, 10^{-2})$	5.8×10^{-4}
N_{ramp}	Number of epochs over which learning rate increases	Uniform	(2, 50)	23
p_{signal}	Probability of batch element containing a signal	Uniform	(0.2, 0.6)	0.277
p_{swap}	Probability of swap augmentation	Uniform	(0, 0.15)	0.014
p_{mute}	Probability of mute augmentation	Uniform	(0, 0.3)	0.055
SNR steps	Number of batches over which SNR scheduler decays	Uniform	(1, 2500)	989

strain windows directly from disk during the training procedure. The length of each noise window sampled from disk is 10.5 s. The first 8 s is used to estimate the power spectral density (PSD) used for whitening. The remaining 2.5 s of the window is whitened in the frequency domain and transformed back to time domain. Due to whitening filter settle-in, 0.5 s of data is corrupted on both ends of the window and removed. Thus, only 1.5 s of data is actually analyzed by the neural network. The PSD estimation, filter construction, and whitening are all done with PyTorch [53] modules to enable GPU-accelerated computation [28]. We use a training batch size of 384, which was chosen such that we fully utilize the GPU memory available. Our out-of-memory data loading is sufficiently fast to support these batch sizes without bottlenecking the preprocessing or neural-network modules.

Noise instances are sampled independently in time from each interferometer. Thus, a noise instance from one interferometer can be paired with many different instances from the other interferometer. This combinatorially increases the amount of unique two-detector noise instances available for optimizing the network. Next, each noise instance has probability p_{invert} to be inverted $[h(t) \to -h(t)]$ and, independently, probability p_{reverse} to be reversed $[h(t) \to h(-t)]$ [54]. Again, the inversion and reversal augmentations increase the amount of unique noise instances in our training data. For transient noise, these augmentations increase the variety of morphologies provided during training, allowing for better generalization to unseen testing data. We fix p_{invert} and p_{reverse} to 0.5.

2. Signal injection

Once a batch of noise instances is generated, simulated BBH signals are added into each 2.5 s unwhitened window with probability $p_{\mathrm{signal}} = 0.277$ and labeled as signals; this signal probability is one of six hyperparameters that we search over (see Table II and the discussion of hyperparameters below). The procedure for injecting signals is as follows: first, intrinsic polarization time series are randomly sampled from the training waveform bank. Next, random extrinsic parameters (right ascension, declination, polarization angle, and SNR) are sampled. The first three of these are sampled from the priors described in Table I; we will discuss the method of SNR sampling in the following paragraph. Intrinsic polarization time series are then projected onto the interferometers and rescaled to the sampled SNR. Randomly sampling extrinsic parameters at training time allows each intrinsic time series to be injected from a variety of sky localizations and distances throughout the training procedure. We found that standard CPU implementations of projecting intrinsic polarizations onto interferometers created bottlenecks that severely limited utilization of GPU resources. We eliminated this bottleneck by developing a PyTorch [53] implementation so that projection can be accelerated using GPUs by a factor of ~200. Finally, the interferometer responses are added into the noise instances. The coalescence time of the merger is randomly placed so that it falls at least 0.25 s from either edge of the 1.5 s whitened noise instance. We enforce this padding because we found that having the coalescence point too close to the left edge of the window makes it more difficult for the neural network to learn, since much of the signal SNR would lie outside the window. The random placement of the coalescence time encodes time translational invariance so that the neural network can identify signals with the coalescence time at different locations throughout the window.

V. TRAINING

A. Curriculum learning

Curriculum learning is a technique for training machine-learning models in which initially, easy to learn samples are provided as training data, and progressively harder samples are introduced over time. One way to apply this in the context of GW detection is to initially provide high SNR signals and gradually introduce lower SNR signals [28]. This allows the neural network to quickly arrive at a minima of its parameter space before trying to optimize for the more realistic task. We begin with an SNR distribution that follows a power law, $p(\mathrm{SNR}) \sim (\mathrm{SNR})^{-3}$, with a minimum of $\mathrm{SNR}_{\mathrm{min}} = 12$ and a maximum of $\mathrm{SNR}_{\mathrm{max}} = 100$. The form of this distribution was chosen to roughly match the SNR distribution of our astrophysically motivated prior. Each time a new training batch is constructed, the minimum SNR bound of the distribution is decreased until we reach the ultimate lower bound of 4. This decrease happens uniformly over 989 batches, a value that was reached through a hyperparameter search.

B. Glitch mitigation

Non-Gaussian noise transients, known as "glitches," can often mimic BBH signals and lead to high-significance false alarms. We implement two types of augmentations we call waveform *muting* and *swapping* to mitigate the impact of transient glitches. These augmentations respectively encode the concepts of coincidence and coherence that true astrophysical signals are expected to exhibit. The values of the parameters controlling these augmentations were determined by hyperparameter search; see below for more details.

1. Muting

For a fraction $p_{\mathrm{mute}} = 0.055$ of the training batch, we inject a BBH signal into only one of the interferometers and label these samples as noise. This teaches the neural network that it is not enough for a BBH-like signal to be present in just one interferometer: coincidence between interferometers is a requirement for true astrophysical signals.

2. Swapping

For an independent fraction of the training batch, $p_{\mathrm{swap}} = 0.014$, we swap one of the interferometer responses with an interferometer response from different signal and label these samples as noise. Thus, these windows will contain BBH waveforms with different intrinsic parameters in each interferometer. This motivates the neural network to learn the concept of coherence: the time-frequency evolution of the signal must be identical in both interferometers.

C. Validation

We construct our validation procedure with the goal of establishing a strong correlation between validation and test metrics. This allows us to confidently pick the best performing neural network during a hyperparameter search, as well as during individual training runs. To accomplish this, our validation procedure is designed to mimic the testing procedure as closely as possible. We reserve 15 000 s of strain data from immediately after the training period and 25 000 waveforms exclusively for neural-network validation during training. These data are not used at all for training the neural network. This temporal choice of training and validation split mimics the real-time production setting, where a deployed neural network is only trained on past data.

To construct our validation set, we first create time slides of the background data until at least 16 hours of live time is accumulated. Similarly to training, these data are batched into 10.5 s windows, with the first 8 s used for whitening the final 2.5 s of each window. As with the training data, 0.5 s of data is cropped from each edge of the window after whitening. Next, we create a dataset of injections by adding waveforms from the validation waveform dataset into the background windows. We set a minimum detector-network SNR threshold of 4 for validation signals. Signals that are quieter are rescaled to the SNR 4 threshold. The SNR is computed with respect to the PSD calculated from the first 8 s of the window. This rescaling procedure mimics the SNR-based rejection sampling performed for the testing dataset. We create five unique injection sets that have the coalescence point of each waveform at 0.25, 0.5, 0.75, 1.0, and 1.25 s within each whitened window. This ensures the validation metric covers a wider variety of scenarios.

The neural network outputs a prediction for each window in the background and injection datasets. We use these predictions to calculate the area under the receiver operating characteristic (AUROC) up to a false positive rate of 10^{-3}, which is the final validation metric. We make this cut on the AUROC so that we are optimizing performance in the regime of low FARs. After the neural-network training has converged, the weights corresponding to the epoch with the highest validation score are used for testing.

D. Hyperparameter search

The hyperparameters of our algorithm are optimized via a random search [55]. It is infeasible to search over all possible hyperparameters, so we selected those that we *a priori* expect to have the greatest impact on the neural-network optimization process. These were the neural network's maximum learning rate (lr_{max}), the number of epochs over which the learning rate "ramps up" (N_{ramp}) to lr_{max}, p_{signal}, p_{mute}, p_{swap}, and the number of steps over which SNR curriculum learning was performed. The priors on each of these parameters can be found in Table II. 30 combinations of these parameters were randomly sampled and used to train a neural network. Of these, the neural network that reported the highest validation score was selected as the neural network used for testing. The hyperparameters used to train this neural network are reported in Table II.

VI. INFERENCE

Our inference pipeline is an ensemble of three models: a snapshotter [41], a whitener, and the neural network itself. Clients send streaming updates of strain data to a snapshotter. The snapshotter sends the latest state to the whitening module. Finally, batches of whitened data are constructed and analyzed by Aframe, producing predictions. The length of the state maintained by the

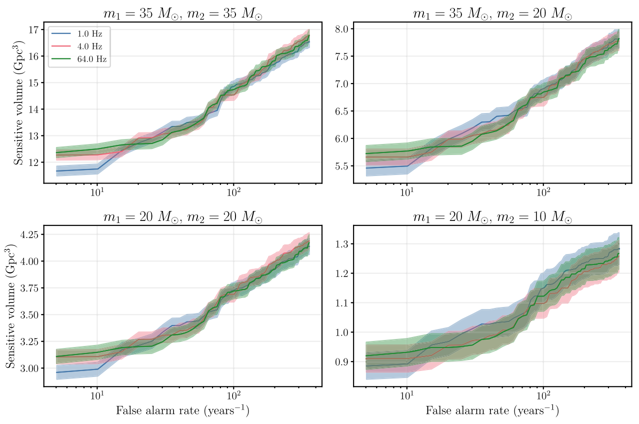

FIG. 2. Sensitivity comparisons for the same neural-network run over the same data at different inference rates. For the purposes of clarity, only a subset of the tested rates are shown here. Except for the 1 Hz inference, all results are within error of each other for all mass combinations and FARs, including for rates not shown in this plot.

snapshotter is determined by the length of the time series used to estimate the PSD, the batch size, and the inference sampling rate. For our analysis, the whitening module uses the first 64 seconds of the snapshotter state to estimate the PSD and build a whitening filter. The remaining data are whitened, and half a second is cropped from both edges to remove the effects of filter settle-in. The whitened data are then unfolded into a batch of overlapping windows. We use a batch size of 128 windows, and, as an inference sampling rate of 4 Hz was used, each 1.5 s window overlaps its neighbors by 1.25 s. This batch of windows is passed to the neural network for prediction. Lastly, neural-network predictions are aggregated client side and postprocessed via the integration and clustering described above.

A critical parameter is the inference sampling rate. The inference sampling rate controls the stride between consecutive windows seen by the neural network. Too small of an inference sampling rate, and astrophysical events may be skipped over. Too large, and computing resources are wasted on redundant inferences. We examined the impact of the inference sampling rate on our sensitivity by repeating trials of our inference procedure at inference sampling rates of 1, 2, 4, 8, 16, and 64 Hz. For this analysis, we accumulated two months worth of time slide data for each trial. Figure 2 shows a subset of the results of this analysis. Algorithms mostly perform within their statistical error. However, at low FARs the 1 Hz analysis has a small performance dip in the 35-35 mass bin. Because analyses performed at 4 Hz require 16 times fewer inference requests than 64 Hz without sacrificing performance, we use an inference sampling rate of 4 Hz for the analyses in this paper.

VII. ALGORITHM LONGEVITY

Noise in gravitational-wave interferometers is nonstationary. Therefore, the timescale over which a single trained neural network will maintain its originally measured performance needs to be evaluated. Determining this timescale helps inform the cadence at which retraining is needed, if at all. To test the longevity of our algorithm, we construct several testing datasets at various intervals across O3. For each interval, we analyze the testing dataset with a

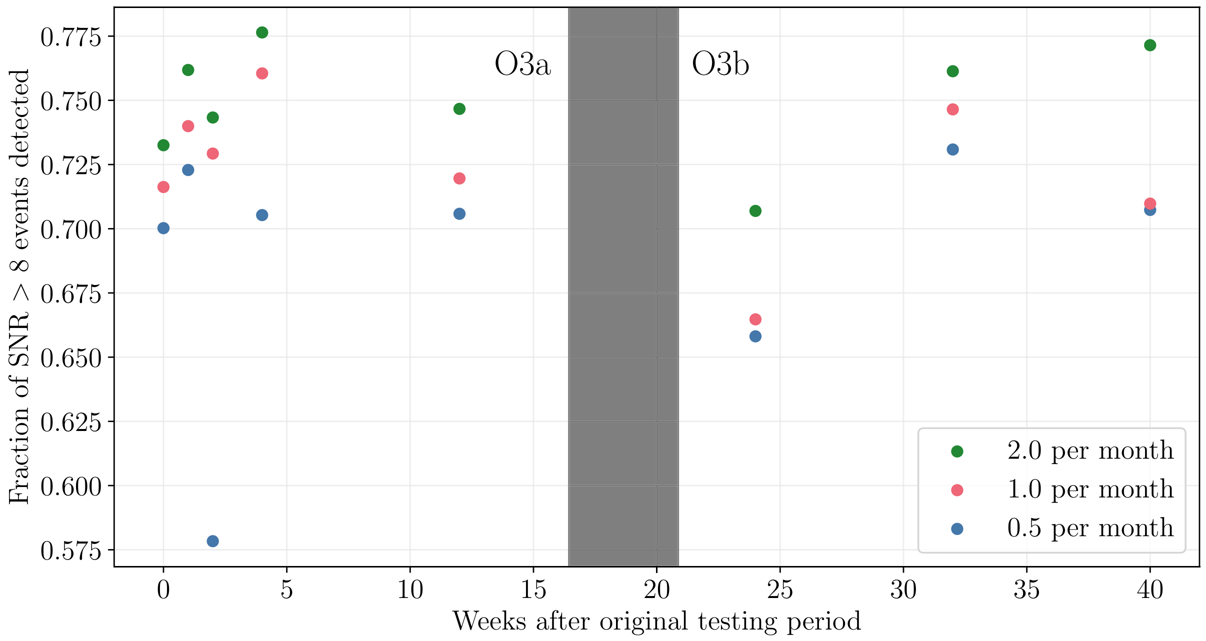

FIG. 3. The fraction of SNR > 8 events detected at different false alarm rates during various weeks across a period of time during O3, beginning May 9th, 2019 and ending March 21st, 2020. Errors on detection fraction estimates are smaller than the plotted points.

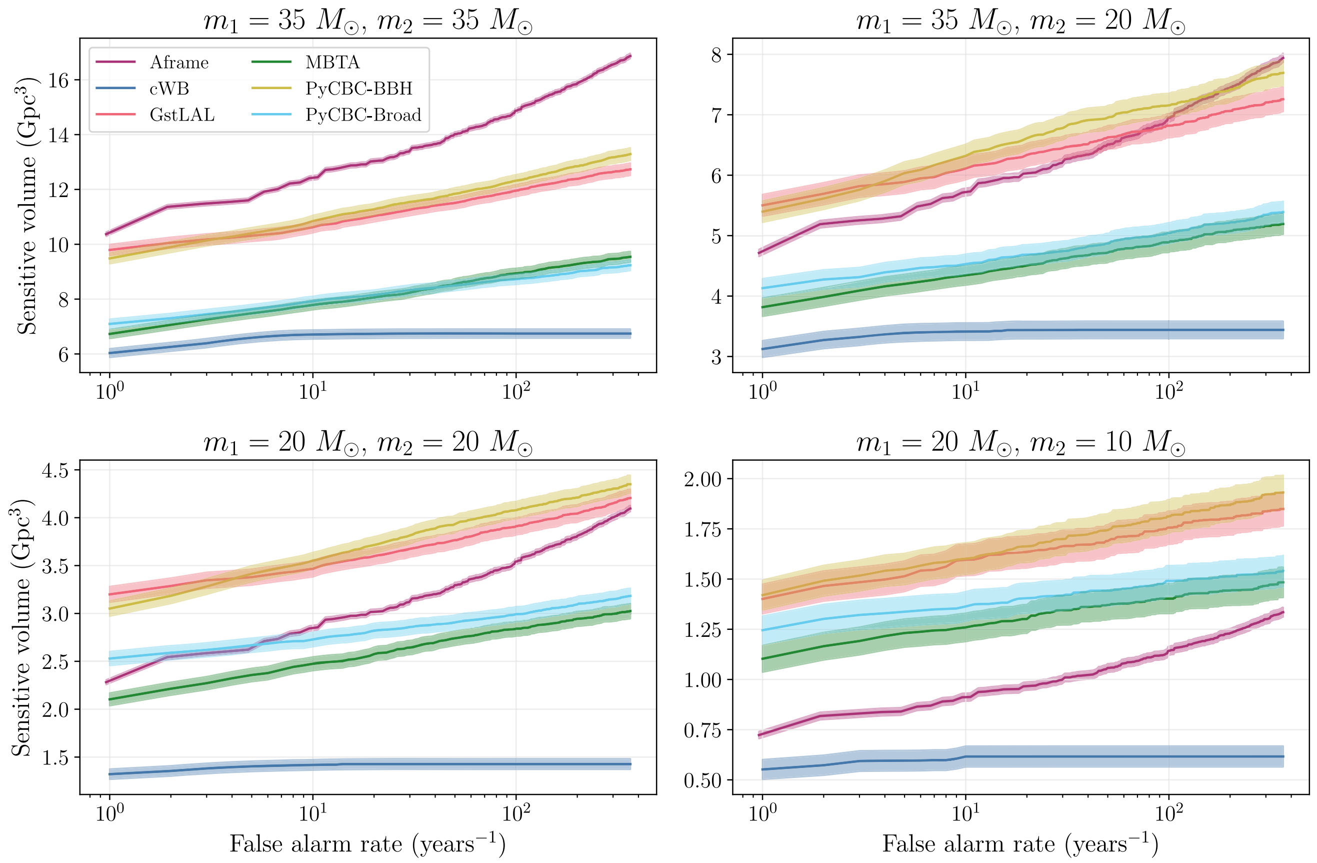

FIG. 4. Sensitive volume vs FAR for four different mass distributions. Masses are specified in the source frame. Each mass is drawn from a log-normal distribution with a mean of the value given above each plot and a width of 0.1. Aframe demonstrates state-of-the-art sensitivity at higher masses but loses performance relative to traditional search pipelines at lower masses. The sensitive volume of the other pipelines was calculated using data from a GWTC-3 data release [52].

neural network trained using the first 10 days of O3 data. This is the same neural network used to produce the results in Fig. 4. To separate the sensitivity of the neural network from the sensitivity of the detectors, we do not measure sensitive volume but instead look at the fraction of events with SNR > 8 that are detected at different FARs. We choose this SNR threshold as it is a typical value often used in GW literature for detectable sources. This metric takes into account the variation in noise level across different time periods, though it does not account for all aspects of detector performance, such as the rate or morphology of glitches. At a FAR of one event per 2 months, a threshold comparable to the one event per 5 months used for releasing significant public alerts by the LVK [56], we see in Fig. 3 that the fractional detection rate of the original neural network does not decay with time. We note that the most significant noise event across all weeks is found during week 2, corresponding to the sharp drop in detection fraction at a FAR of 1 per 2 months. Though there is some fluctuation from week to week, a single neural network trained on a week's worth of data at the beginning of the observing run maintains sensitivity over the duration of the run.

VIII. LATENCY AND COMPUTATIONAL REQUIREMENTS

Training the neural network with a single NVIDIA 16 GB Tesla V100 GPU takes approximately 43 h, and once trained, the neural network can continue to be used for months without retraining; see the discussion of algorithm longevity in Sec. V for details. For inference, we utilize a Triton inference server [42] that is hosted on a NVIDIA DGX server containing eight 16 GB Tesla V100 GPUs (see Sec. VI for details on inference configuration). Altogether, analyzing the one year of background data and one year of injections used in this analysis to create Fig. 4 takes approximately 4 h, corresponding to a throughput of about 500 s of data from a two-detector network analyzed per second per GPU. Normalizing by the quantity of GPUs and number of interferometers in the analysis configuration, this corresponds to an order of magnitude improvement in throughput compared with previous work by Huerta *et al.* [43], a factor of ∼2.5 compared with Chaturvedi *et al.* [57] and a factor of ∼1.5 improvement compared with Tian *et al.* [58]. This improvement is due to the use of a more efficient neural-network architecture, as well as the IaaS model described above.

TABLE III. Sources of latency for an Aframe online analysis. For the items listed in the upper section this table, the latency does not come from performing the computation, but rather from needing to wait for the data to exist before the action can occur. Items in the lower section are computational steps, and we report the median timing of 9191 trials. The upper and lower error bars represent the 95th and 5th percentile, respectively. All measurements were taken on a dedicated NVIDIA A30 GPU.

Latency source	Latency (s)
Coalescence point exiting training kernel padding	0.25
Cropping corruption from whitening filter	0.50
Cropping corruption from resampling to 2048 Hz	1.0
Integrating network output	1.0
Reading data and transferring to GPU	$1.03^{+0.06}_{-0.05} \times 10^{-2}$
Estimating PSD and whitening	$8.77^{+1.35}_{-0.31} \times 10^{-4}$
Performing inference on whitened data	$9.63^{+0.38}_{-0.32} \times 10^{-3}$
Integrating and aggregating network output	$3.42^{+0.02}_{-0.01} \times 10^{-1}$
Identifying candidate events in integrated output	$1.40^{+0.62}_{-0.43} \times 10^{-4}$
Total	$3.114^{0.006}_{0.001}$

With trained neural-network weights in hand, the requirements for online deployment are much smaller. A single NVIDIA 24 GB A30 GPU is sufficient for real-time inference at an inference sampling rate of 2048 Hz, which provides sufficient resolution for coalescence time estimation. The total memory required to hold both the neural network and data is 4.6 GB. The computational latency of the neural network is less than 10 ms. In practice, the latency of our algorithm is dominated by pre- and post-processing steps that bring the total latency to approximately 3.1 s. For a detailed accounting of sources of latency within Aframe, see Table III. The most significant source of latency in an online analysis comes from waiting for data to exist such that data can be cropped from the edges after resampling and whitening. All other computational steps (data reading and writing, data transfer to and from GPU, whitening, event identification, etc.) take less than 0.4 s combined. In production, additional latency is incurred uploading events to the Gravitational-wave Candidate Event Database (GraceDB) [59]. This latency is not included in this 3.1 s estimate. A recent study [60] used a real-time mock data challenge replay of O3 data to benchmark pipeline latencies, including GraceDB processing. Analyzing this data stream, we find a median (90%) event reporting latency of 3.9 s (4.3 s), in good agreement with our latency budget. Matched filtering pipelines report a median (90%) latency of 12.3 s (41.4 s).

IX. COMPARISON WITH EXISTING SEARCHES

To demonstrate Aframe's readiness for real-time deployment, we compare its sensitivity to search pipelines used in production by the LVK. For our pipeline, estimating sensitive volume requires analyzing simulated GW events "injected" into strain data and analyzing background live time produced by "time slides." Performing time slides is a standard way of empirically estimating the background (i.e. the distribution of noise events) for a search pipeline which analyzes a network of detectors. In brief, the strain from one detector is shifted in time by an amount greater than the gravitational-wave travel time between the detectors ($\sim$10 ms for the two LIGO detectors). Therefore, any reported triggers could not have been caused by an astrophysical event. In this analysis, the Hanford strain data are held fixed and the Livingston data are shifted in 1 s increments until the required background live time is accumulated. Then, a FAR can be assigned to injected events by dividing the number of noise events with detection statistic greater than the event of interest, with live time analyzed. All GW detections reported in the third Gravitational-Wave Transient Catalog (GWTC-3) [2] were excised from the background.

A useful metric to measure the sensitivity of search algorithms is the *sensitive volume*. Sensitive volume measures the volume over which some astrophysical population of sources distributed uniformly in comoving volume is detectable at a given FAR. Sensitive volume was used to measure the sensitivity of search pipelines in GWTC-3. This provides an astrophysically meaningful benchmark to compare the performance of Aframe to the performance of traditional searches. More details on the sensitive volume calculation can be found in Sec. III. Figure 4 compares Aframe's sensitive volume as a function of FAR with the sensitivity of the MBTA [61], PyCBC [62], GstLAL [63,64] and cWB [65] searches as reported in GWTC-3 [2]. We note that the template banks used by MBTA, GstLAL, and PyCBC-Broad in the GWTC-3 analysis contain waveforms outside of the $5 - 100 M_\odot$ range searched by Aframe. In principle, these searches could increase their sensitivities in the $5 - 100 M_\odot$ range by

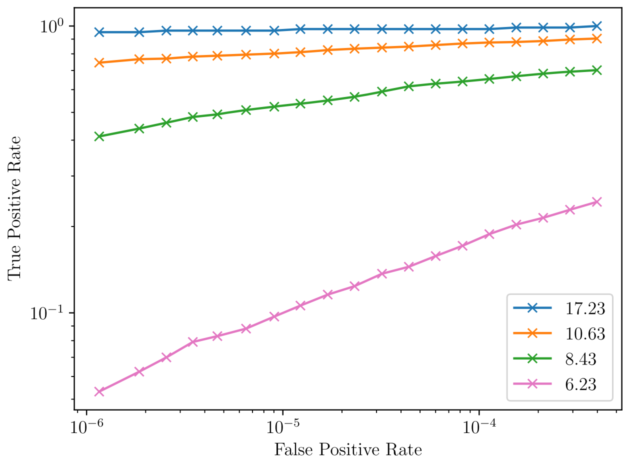

FIG. 5. ROC curves for waveforms in different SNR bins in our testing dataset, described in Sec. IV. Each bin contains waveforms with SNRs within 0.01 of the given value.

removing these templates. This is evident when comparing the performance of PyCBC-BBH and PyCBC-Broad in Fig. 4. For the future, we encourage production-level LVK CBC pipelines to publish BBH-specific sensitivities against which developing ML pipelines can benchmark.

In the $35-35M_\odot$ mass distribution, Aframe has a larger sensitive volume than the GWTC-3 configurations of all searches and is comparable in the $35-20M_\odot$ mass bin, for the FARs considered in this analysis. As source masses decrease further, so does Aframe's performance relative to existing pipelines. This is in part due to our neural-network architectures inability to model the longer-duration features of these low-mass signals. While the architecture implements global pooling layers, the convolution layers use a kernel length of three samples. Improvements to neural-network architecture design such as utilizing dilated convolutions that can better model these features will help to improve performance at these mass ranges.

Previous studies of ML-based gravitational-wave detection algorithms tend not to use sensitive volume as a metric, preferring instead to use traditional ML metrics such as ROC curves (an exception is [66], which uses a non-astrophysical prior and a Euclidean volume distribution). This makes direct comparison difficult, as these metrics depend on the parameter distributions of tested events. Still, in Fig. 5 we present our own ROC curve to directly compare to metrics published in previous works [43,57]. We report a higher true positive rate (TPR) for all SNR thresholds at the lowest false positive rate of $\sim 10^{-6}$ compared to these previous studies. This is particularly true for the lowest SNR threshold of 6.23, where we achieve 2 orders of magnitude of improvement (a TPR of 0.053, compared to approximately 5×10^{-4}). However, we encourage future studies to use sensitive volume to astrophysically motivated distributions as the measure of performance.

X. DETECTING ASTROPHYSICAL CANDIDATES IN GWTC-3

The testing period we use contains nine astrophysical candidate events reported as significant detections in GWTC-3. While we evaluated our algorithm's performance using "time slides" of these data (see Sec. IX), we also analyzed the unshifted (or "zero-lag") data to determine if our algorithm detects these known candidates. The results of this analysis are shown in Table IV. We detect all nine candidates, with eight of the nine candidates detected at a false alarm rate of less than 1 per year, the minimum possible value for this analysis. For the final event, our reported false alarm rate, 14 per year, is of a similar magnitude to the false alarm rate reported by the GWTC-3 pipelines at 2.8 per year. Additionally, during this period, we do not report any noncatalog candidates with a false alarm rate less than 5 per month.

TABLE IV. Masses in units of $M_\odot$ and false alarm rates in units of inverse years from Aframe, cWB, GstLAL, MBTA, and PyCBC-BBH for the known events in our testing set. Masses come from Table VIII of GWTC-2.1 [33] and FARs from Table XV of GWTC-3 [2]. As our analysis examined only one year of background, our minimum FAR is one per year.

Event	$m_1(M_\odot)$	$m_2(M_\odot)$	Aframe	cWB	GstLAL	MBTA	PyCBC-BBH	PyCBC-Broad
GW190512_180714	$23.2^{+5.6}_{-5.6}$	$12.5^{+3.5}_{-2.6}$	< 0.97	0.88	$< 1.0\times10^{-5}$	0.038	$< 1.1\times10^{-4}$	1.1×10^{-4}
GW190513_205428	$36.0^{+10.6}_{-9.7}$	$18.3^{+7.4}_{-4.7}$	< 0.97	⋯	1.3×10^{-5}	0.11	0.044	19
GW190514_065416	$40.9^{+17.3}_{-9.3}$	$28.4^{+10.0}_{-10.1}$	14	⋯	450	⋯	2.8	⋯
GW190517_055101	$39.2^{+13.9}_{-9.2}$	$24.0^{+7.4}_{-7.9}$	< 0.97	0.0065	0.0045	0.11	3.5×10^{-4}	0.0095
GW190519_153544	$65.1^{+10.8}_{-11.0}$	$40.8^{+11.5}_{-12.7}$	< 0.97	3.1×10^{-4}	$< 1.0\times10^{-5}$	7.0×10^{-5}	$< 1.1\times10^{-4}$	$< 1.0\times10^{-4}$
GW190521	$98.4^{+33.6}_{-21.7}$	$57.2^{+27.1}_{-30.1}$	< 0.97	2.0×10^{-4}	0.20	0.042	0.0013	0.44
GW190521_074359	$43.4^{+5.8}_{-5.5}$	$33.4^{+5.2}_{-6.8}$	< 0.97	1.0×10^{-4}	$< 1.0\times10^{-5}$	1.0×10^{-5}	$< 2.3\times10^{-5}$	$< 1.8\times10^{-5}$
GW190527_092055	$35.6^{+18.7}_{-8.0}$	$22.2^{+9.0}_{-8.7}$	< 0.97	⋯	0.23	⋯	19	⋯
GW190602_175927	$71.8^{+18.1}_{-14.6}$	$44.8^{+15.5}_{-19.6}$	< 0.97	0.015	$< 1.0\times10^{-5}$	3.0×10^{-4}	0.013	0.29

XI. CONCLUSION

We have implemented a machine-learning-based CBC search pipeline that is capable of low-latency use in a production setting. Through robust data augmentation techniques and extensive work in developing software infrastructure (Sec. XI), our algorithm achieves a sensitivity that is superior to or competitive with established search pipelines for higher-mass BBHs. Work remains to improve the algorithm's performance on lower-mass BBH systems. We leave these investigations to future work.

There are a number of extensions we plan to investigate in future work. As a means of reducing the scope of this study, our algorithm was trained requiring data from both Hanford and Livingston, limiting analysis to when both of these detectors are online. However, accounting for instances where specific detectors are offline is important for maximizing analysis live time. This could take the form of four-detector, three-detector, pairwise, and single-detector models. Investigating additional detector combinations is left for future work. Further, low-latency alerts are less important for BBHs than BNS and neutron star–black hole mergers, where electromagnetic counterparts are more likely. Many algorithms have already been published on this front [67–71]. However, appreciable sensitivity compared with state-of-the-art matched filtering approaches has yet to be demonstrated. The detection of these mergers with neural networks is more challenging due to the greater length of time these signals spend in the sensitive band of the detector.

Finally, because Aframe does not provide an SNR time series for detections, it cannot be used in conjunction with current low-latency sky localization algorithms like BAYESTAR [72]. However, machine-learning-based parameter estimation algorithms, like AMPLFI [73], are being developed as low-latency solutions to sky localization and can be used to followup Aframe detections.

ACKNOWLEDGMENTS

W. B. and M. W. C. acknowledge support from the National Science Foundation (NSF) with Grants No. PHY-2010970 and No. PHY-2117997. E. M., E. K., A. G., and D. C. acknowledge support from the NSF under Grants No. PHY-1764464 and No. PHY-2309200 to the LIGO Laboratory and No. PHY-2117997 to the A3D3 Institute. E. M., E. K., A. G., and D. C. also acknowledge NSF Grants No. PHY-1931469 and No. PHY-1934700. The authors express sincere gratitude to Reed Essick for clarifying numerous nuances of the sensitive volume calculation. The authors are also very grateful to Thomas Dent for identifying a public dataset which could be used to calculate the performance of traditional search pipelines. Additionally, the authors thank Vassilis Skliris and Becca Ewing for their useful comments and discussion on earlier drafts of this work. This research was undertaken with the support of the LIGO computational clusters. This material is based upon work supported by the NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation. This research has made use of data obtained from the Gravitational Wave Open Science Center, a service of LIGO Laboratory, the LIGO Scientific Collaboration and the Virgo Collaboration.

SOFTWARE AVAILABILITY

All code used to produce results in this work is publicly available. The Aframe project repository can be found at [74].

In addition, two open source libraries, ML4GW [75] and HERMES [76], were developed to support this work. The ML4GW library contains PyTorch utilities for efficient on-GPU data loading, whitening, PSD estimation and other data processing techniques common to GW analysis. The hermes library contains utilities for deploying models in the IaaS paradigm via Triton inference servers.

DATA AVAILABILITY

The data that support the findings of this article are openly available [52].

[1] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. Lett. **116**, 061102 (2016).

[2] R. Abbott, T. Abbott, F. Acernese, K. Ackley, C. Adams, N. Adhikari, R. Adhikari, V. Adya, C. Affeldt, D. Agarwal *et al.*, Phys. Rev. X **13**, 041039 (2023).

[3] J. Aasi, B. P. Abbott, R. Abbott, T. Abbott, M. R. Abernathy, K. Ackley, C. Adams, T. Adams, P. Addesso *et al.* (LIGO Scientific Collaboration), Classical Quantum Gravity **32**, 074001 (2015).

[4] F. Acernese, M. Agathos, K. Agatsuma, D. Aisa, N. Allemandou, A. Allocca, J. Amarni, P. Astone, G. Balestri, G. Ballardin *et al.*, Classical Quantum Gravity **32**, 024001 (2014).

[5] T. Akutsu, M. Ando, K. Arai, Y. Arai, S. Araki, A. Araya, N. Aritomi, Y. Aso, S. Bae, Y. Bae *et al.*, Prog. Theor. Exp. Phys. **2021**, 05A101 (2020).

[6] S. Dwyer, D. Sigg, S. W. Ballmer, L. Barsotti, N. Mavalvala, and M. Evans, Phys. Rev. D **91**, 082001 (2015).

[7] M. Punturo, M. Abernathy, F. Acernese, B. Allen, N. Andersson, K. Arun, F. Barone, B. Barr, M. Barsuglia, M. Beker *et al.*, Classical Quantum Gravity **27**, 194002 (2010).

[8] T. Edwards, DSS Technical Report No. LI-RP-DS-009, 2000.

[9] N. R. Tanvir, A. J. Levan, A. S. Fruchter, J. Hjorth, R. A. Hounsell, K. Wiersema, and R. L. Tunnicliffe, Nature (London) **500**, 547 (2013).

[10] S. Ascenzi, M. W. Coughlin, T. Dietrich, R. J. Foley, E. Ramirez-Ruiz, S. Piranomonte, B. Mockler, A. Murguia-Berthier, C. L. Fryer, N. M. Lloyd-Ronning *et al.*, Mon. Not. R. Astron. Soc. **486**, 672 (2019).

[11] Z.-P. Jin, S. Covino, N.-H. Liao, X. Li, P. D'Avanzo, Y.-Z. Fan, and D.-M. Wei, Nat. Astron. **4**, 77 (2020).

[12] J. C. Rastinejad, B. P. Gompertz, A. J. Levan, W.-f. Fong, M. Nicholl, G. P. Lamb, D. B. Malesani, A. E. Nugent, S. R. Oates, N. R. Tanvir *et al.*, Nature (London) **612**, 223 (2022).

[13] A. Albert, S. Alves, M. André, M. Ardid, S. Ardid, J.-J. Aubert, J. Aublin, B. Baret, S. Basa, Y. Becherini *et al.*, J. Cosmol. Astropart. Phys. 04 (2023) 004.

[14] R. Abbasi, M. Ackermann, J. Adams, N. Aggarwal, J. A. Aguilar, M. Ahlers, M. Ahrens, J. M. Alameddine, A. A. Alves, N. M. Amin *et al.*, Astrophys. J. **944**, 80 (2023).

[15] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. Lett. **119**, 161101 (2017).

[16] A. Goldstein, P. Veres, E. Burns, M. S. Briggs, R. Hamburg, D. Kocevski, C. A. Wilson-Hodge, R. D. Preece, S. Poolakkil, O. J. Roberts *et al.*, Astrophys. J. **848**, L14 (2017).

[17] V. Savchenko, C. Ferrigno, E. Kuulkers, A. Bazzano, E. Bozzo, S. Brandt, J. Chenevez, T. J.-L. Courvoisier, R. Diehl, A. Domingo *et al.*, Astrophys. J. **848**, L15 (2017).

[18] B. P. Abbott *et al.*, Astrophys. J. Lett. **848**, L13 (2017).

[19] V. Savchenko, C. Ferrigno, E. Kuulkers, A. Bazzano, E. Bozzo, S. Brandt, J. Chenevez, T. J.-L. Courvoisier, R. Diehl, A. Domingo *et al.*, Astrophys. J. Lett. **848**, L15 (2017).

[20] D. A. Coulter *et al.*, Science **358**, 1556 (2017).

[21] S. J. Smartt *et al.*, Nature (London) **551**, 75 (2017).

[22] B. P. Abbott *et al.*, Astrophys. J. Lett. **848**, L12 (2017).

[23] P. Harris, E. Katsavounidis, W. P. McCormack, D. Rankin, Y. Feng, A. Gandrakota, C. Herwig, B. Holzman, K. Pedro, N. Tran *et al.*, arXiv:2203.16255.

[24] G. Baltus, J. Janquart, M. Lopez, A. Reza, S. Caudill, and J.-R. Cudell, Phys. Rev. D **103**, 102003 (2021).

[25] C. Verma, A. Reza, G. Gaur, D. Krishnaswamy, and S. Caudill, arXiv:2206.12673.

[26] P. G. Krastev, Phys. Lett. B **803**, 135330 (2020).

[27] D. George and E. A. Huerta, Phys. Rev. D **97**, 044039 (2018).

[28] P. Nousi, A. E. Koloniari, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, Phys. Rev. D **108**, 024022 (2023).

[29] Online pipelines—IGWN | Public alerts user guide 2023, https://emfollow.docs.ligo.org/userguide/analysis/searches.html (2023).

[30] V. Skliris, M. R. K. Norman, and P. J. Sutton, arXiv:2009.14611.

[31] A. Paleyes, R.-G. Urma, and N. D. Lawrence, ACM Comput. Surv. **55**, 1 (2022).

[32] B. P. Abbott, R. Abbott, T. D. Abbott, S. Abraham, F. Acernese, K. Ackley, C. Adams, R. X. Adhikari, V. B. Adya, C. Affeldt *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. X **9**, 031040 (2019).

[33] R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, N. Adhikari, R. X. Adhikari, V. B. Adya, C. Affeldt *et al.* (LIGO Scientific and Virgo Collaborations), arXiv:2108.01045.

[34] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Phys. Rev. D **100**, 063015 (2019).

[35] 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.

[36] S. Ioffe and C. Szegedy, arXiv:1502.03167.

[37] Y. Wu and K. He, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Springer, Cham, 2018).

[38] D. P. Kingma and J. Ba, arXiv:1412.6980.

[39] L. N. Smith and N. Topin, arXiv:1708.07120.

[40] A. Gunny, D. Rankin, J. Krupa, M. Saleem, T. Nguyen, M. Coughlin, P. Harris, E. Katsavounidis, S. Timm, and B. Holzman, Nat. Astron. **6**, 529 (2022).

[41] A. Gunny, D. Rankin, P. Harris, E. Katsavounidis, E. Marx, M. Saleem, M. Coughlin, and W. Benoit, in *Proceedings of the 12th Workshop on AI and Scientific Computing at Scale Using Flexible Computing Infrastructures*, FlexScience '22 (Association for Computing Machinery, New York, 2022), pp. 9–17, ISBN 9781450393096.

[42] NVIDIA Corporation, Triton inference server: An optimized cloud and edge inferencing solution, https://github.com/triton-inference-server/server (2023).

[43] E. A. Huerta, A. Khan, X. Huang, M. Tian, M. Levental, R. Chard, W. Wei, M. Heflin, D. S. Katz, V. Kindratenko *et al.*, Nat. Astron. **5**, 1062 (2021).

[44] M. B. Schäfer, F. Ohme, and A. H. Nitz, Phys. Rev. D **102**, 063015 (2020).

[45] H.-Y. Chen, D. E. Holz, J. Miller, M. Evans, S. Vitale, and J. Creighton, Classical Quantum Gravity **38**, 055010 (2021).

[46] V. Tiwari, Classical Quantum Gravity **35**, 145009 (2018).

[47] W. M. Farr, Res. Not. Am. Astron. Soc. **3**, 66 (2019).

[48] R. Abbott, H. Abe, F. Acernese, K. Ackley, S. Adhicary, N. Adhikari, R. X. Adhikari, V. K. Adkins, V. B. Adya *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), Astrophys. J. Suppl. Ser. **267**, 29 (2023).

[49] G. Ashton, M. Hübner, P. D. Lasky, C. Talbot, K. Ackley, S. Biscoveanu, Q. Chu, A. Divakarla, P. J. Easter, B. Goncharov *et al.*, Astrophys. J. Suppl. Ser. **241**, 27 (2019).

[50] M. Hannam, P. Schmidt, A. Bohé, L. Haegel, S. Husa, F. Ohme, G. Pratten, and M. Pürrer, Phys. Rev. Lett. **113**, 151101 (2014).

[51] R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, N. Adhikari, R. X. Adhikari, V. B. Adya, C. Affeldt, D. Agarwal *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), Phys. Rev. X **13**, 011048 (2023).

[52] LIGO Scientific, Virgo, and KAGRA Collaborations, GWTC-3: Compact binary coalescences observed by LIGO and Virgo during the second part of the third observing run—O3 search sensitivity estimates, 10.5281/zenodo.7890437 (2023).

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*,

PyTorch: An Imperative Style, High-Performance Deep Learning Library (Curran Associates Inc., Red Hook, NY, 2019).

[54] S. Bini, G. Vedovato, M. Drago, F. Salemi, and G. A. Prodi, Classical Quantum Gravity **40**, 135008 (2023).

[55] J. Bergstra and Y. Bengio, J. Mach. Learn. Res. **13**, 281 (2012), https://dl.acm.org/doi/10.5555/2188385.2188395.

[56] https://emfollow.docs.ligo.org/userguide/analysis/

[57] P. Chaturvedi, A. Khan, M. Tian, E. A. Huerta, and H. Zheng, Front. Artif. Intell. **5**, 828672 (2022).

[58] M. Tian, E. A. Huerta, H. Zheng, and P. Kumar, Mach. Learn. **5**, 025056 (2024).

[59] https://gracedb.ligo.org/

[60] S. S. Chaudhary, A. Toivonen, G. Waratkar, G. Mo, D. Chatterjee, S. Antier, P. Brockill, M. W. Coughlin, R. Essick, S. Ghosh *et al.*, arXiv:2308.04545.

[61] F. Aubin, F. Brighenti, R. Chierici, D. Estevez, G. Greco, G. M. Guidi, V. Juste, F. Marion, B. Mours, E. Nitoglia *et al.*, Classical Quantum Gravity **38**, 095004 (2021).

[62] T. D. Canton, A. H. Nitz, B. Gadre, G. S. C. Davies, V. Villa-Ortega, T. Dent, I. Harry, and L. Xiao, Astrophys. J. **923**, 254 (2021).

[63] B. Ewing, R. Huxford, D. Singh, L. Tsukada, C. Hanna, Y.-J. Huang, P. Joshi, A. K. Y. Li, R. Magee, C. Messick *et al.*, arXiv:2305.05625.

[64] L. Tsukada, P. Joshi, S. Adhicary, R. George, A. Guimaraes, C. Hanna, R. Magee, A. Zimmerman, P. Baral, A. Baylor *et al.*, Phys. Rev. D **108**, 043004 (2023).

[65] M. Drago, S. Klimenko, C. Lazzaro, E. Milotti, G. Mitselmakher, V. Necula, B. O'Brian, G. A. Prodi, F. Salemi, M. Szczepanczyk *et al.*, SoftwareX **14**, 100678 (2021).

[66] M. B. Schäfer, O. Zelenka, A. H. Nitz, H. Wang, S. Wu, Z.-K. Guo, Z. Cao, Z. Ren, P. Nousi, N. Stergioulas *et al.*, Phys. Rev. D **107**, 023021 (2023).

[67] B.-J. Lin, X.-R. Li, and W.-L. Yu, Front. Phys. **15** (2019).

[68] M. B. Schäfer, F. Ohme, and A. H. Nitz, Phys. Rev. D **102** (2020).

[69] P. G. Krastev, arXiv:1908.03151.

[70] J. Aveiro, F. F. Freitas, M. Ferreira, A. Onofre, C. Providência, G. Gonçalves, and J. A. Font, Phys. Rev. D **106** (2022).

[71] R. Qiu, P. G. Krastev, K. Gill, and E. Berger, Phys. Lett. B **840**, 137850 (2023).

[72] L. P. Singer and L. R. Price, Phys. Rev. D **93** (2016).

[73] D. Chatterjee, E. Marx, W. Benoit, R. Kumar, M. Desai, E. Govorkova, A. Gunny, E. Moreno, R. Omer, R. Raikman *et al.*, arXiv:2407.19048.

[74] https://github.com/ML4GW/aframe

[75] https://github.com/ML4GW/ml4gw

[76] https://github.com/ML4GW/hermes