Convolutional neural networks for signal detection in real LIGO data

Ondřej Zelenka,[1,2,3,*] Bernd Brügmann,[1,2] and Frank Ohme[4,5]
[1]*Friedrich-Schiller-Universität Jena, D-07743 Jena, Germany*
[2]*Michael Stifel Center Jena, D-07743 Jena, Germany*
[3]*Astronomical Institute of the Czech Academy of Sciences, Boční II 1401/1a, CZ-141 00 Prague, Czech Republic*
[4]*Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut, D-30167 Hannover, Germany*
[5]*Leibniz Universität Hannover, D-30167 Hannover, Germany*

(Received 21 February 2024; accepted 14 May 2024; published 10 July 2024)

Searching the data of gravitational-wave detectors for signals from compact binary mergers is a computationally demanding task. Recently, machine-learning algorithms have been proposed to address current and future challenges. However, the results of these publications often differ greatly due to differing choices in the evaluation procedure. The Machine Learning Gravitational-Wave Search Challenge was organized to resolve these issues and produce a unified framework for machine-learning search evaluation. Six teams submitted contributions, four of which are based on machine-learning methods, and two are state-of-the-art production analyses. This paper describes the submission from the team TPI FSU Jena and its updated variant. We also apply our algorithm to real O3b data and recover the relevant events of the GWTC-3 catalog.

DOI: 10.1103/PhysRevD.110.024024

I. INTRODUCTION

One of the most powerful known sources of gravitational waves (GWs) is a compact binary coalescence: the final stage of a binary composed of compact objects, such as a black hole or neutron star. Analyzing the signal from such an event allows us to constrain the source parameters, such as component masses, which greatly contributes to the study of the black hole population in the Universe, and the mechanism by which supermassive black holes are formed [1,2]. For this reason, GW observations are crucial in expanding our understanding of the Universe.

Most contemporary detection pipelines are based on *matched filtering* [3] and use a *template bank* of expected waveforms. These pipelines are highly sensitive to signals covered by the template bank, but less sensitive to others. *Loosely modeled searches* are a complementary approach: they do not require the advanced knowledge of waveforms to be searched for, but they are less sensitive to compact-binary mergers than matched-filter searches [4–6].

With the broadening of the sensitive frequency range of detectors, it becomes necessary to increase the density of template banks. In addition, expanding the parameter space of interest typically requires more templates to cover the signal manifold. This causes a steep rise in the size of template banks and therefore computational time of matched-filtering-based algorithms. In particular, this is an issue when incorporating effects such as eccentricity [7], precession [8,9], or higher-order modes [9,10].

Moreover, matched-filter searches are optimal for an idealized Gaussian noise distribution. However, actual detector data deviate from this assumption [11]. While measures are taken to reduce the effect of this deviation, there are still optimizations to be made. These are some of the driving forces behind the search for new, more efficient methods to complement the matched-filter-based analyses.

A rather new development is using machine learning (ML) methods in GW astronomy. This was started by two pioneering papers on the topic of GW detection [12,13]. Their approach consisted of applying convolutional neural networks to recognize whether individual one second-long whitened samples of Gaussian noise contain a binary black hole (BBH) GW signal. In another direction, applications in parameter estimation [14–16], denoising [17,18], fast waveform generation [19], and more [20] have also been published; we, however, remain focused on the detection problem in this paper.

In the recent years, a multitude of new results have been achieved on this topic [20–23]. However, owing to differing choices in the generation of test data, results in the literature are difficult to compare to each other. To resolve this issue, the Machine Learning Gravitational-Wave Search Challenge

*ondrej.zelenka@asu.cas.cz

(MLGWSC-1) [24,25] has been organized. From 12 October 2021 until 14 April 2022, multiple teams developed ML-based algorithms for the detection of GW signals originating in BBH mergers in month-long streams of data from the two U.S.-based Laser Interferometer Gravitational Observatory (LIGO) detectors [26]. The final test data were unknown to participants but followed a known distribution, and no scoreboard was kept during the challenge. Eventually, four ML-based submissions were received, as well as two conventional algorithms to provide a baseline. Their performance has been evaluated in detail, and effects responsible for the differing performance of submissions have been isolated.

We have authored one of the challenge submissions, titled "TPI FSU Jena." On test datasets following a simplified Gaussian noise distribution, our search was the top ML submission and performed close to the matched-filter baseline, a similar submission being a close second. In addition, it had a comparatively short runtime. However, on test data generated using LIGO open data [27], non-Gaussian noise artifacts polluted the search results to a large degree.

In this work, we first briefly describe the MLGWSC-1, our submission, and choices made during its development. Following that, we describe the steps taken to further optimize the contribution after the end of the challenge, which greatly improve its performance when non-Gaussian noise transients are present in the data. Finally, we demonstrate the power of the developed searches by applying them to open data from the second half of the third observing run and recovering the events of the third Gravitational-Wave Transient Catalog (GWTC-3) lying in the relevant portion of the source parameter space.

II. MLGWSC-1

A. Test data

The test data consist of two strains from the LIGO Hanford and Livingston detectors. The script used to generate them was available to participants of the challenge with the option to specify its seed. For the final evaluation, a challenge dataset in the length of one month was generated after the challenge deadline using a previously unknown seed [24].

The test data exist in four levels named *datasets* of progressively increasing difficulty. The first three use background noise generated by a colored Gaussian model, while the fourth uses real noise from the O3a observing run [27]. The injection complexity is also increasing, from nonspinning, dominant mode only, to precessing waveforms with generic misaligned spins and multiple higher-order modes.

Test data are generated using the script `generate_data.py` supplied by the MLGWSC-1 [25], which creates the background noise, generates waveforms, and injects them into the noise, forming the foreground. Both the background and the foreground are stored in HDF5 files [28], each containing groups titled L1 and H1 for the Livingston and Hanford detectors, with the full length of the strain split into multiple segments labeled by their GPS start time. These segments are generated independently of each other. All time series are sampled at a rate of 2048 Hz. A low frequency cutoff of 15 Hz is applied to the background noise to allow for reduction in data size of the real detector noise to be downloaded.

The injection parameters are generated by the astrophysical distribution for all angular parameters, and the distance is specified by generating the chirp distance, defined as [29]

$$d_c = d \cdot \left(\frac{\mathcal{M}_{c,0}}{\mathcal{M}_c}\right)^{5/6}, \tag{1}$$

where d is the luminosity distance, $\mathcal{M}_c = (m_1 m_2)^{3/5}/(m_1+m_2)^{1/5}$ is the chirp mass, and $\mathcal{M}_{c,0} = 1.4/2^{1/5} M_\odot$ is a fiducial chirp mass. The squared chirp distance is drawn from a uniform distribution over the interval $d_c^2 \in [130^2\ \mathrm{Mpc}^2, 350^2\ \mathrm{Mpc}^2]$. Component masses are drawn in the detector frame from uniform distributions over different intervals depending on the dataset, following the primary/secondary mass constraint $m_1 \geq m_2$.

The events are placed at random intervals between 24 s and 30 s between merger times. The waveforms are generated using the `IMRPhenomXPHM` [30] phenomenological model, capable of accurate modeling of precession and higher-order modes. They are then projected on the corresponding detectors and injected into the background data to produce the foreground.

In the first dataset, Gaussian noise is generated, from the `aLIGOZeroDetHighPower` power spectral density (PSD) [31,32] (see Sec. III A). Component masses $m_1, m_2 \in [10M_\odot, 50M_\odot]$ are drawn from a uniform distribution, all six spin components are set to zero, only the dominant $2, \pm 2$ modes are used, and the low frequency cutoff is chosen to be 20 Hz.

In the second dataset, Gaussian noise is generated using an unknown PSD. From a set of 20 PSDs derived from the O3a observing run data [27], for each detector, one is randomly chosen and used to generate the noise in all segments. Component masses $m_1, m_2 \in [7M_\odot, 50M_\odot]$ are drawn from a uniform distribution, and both spins are aligned with the orbital angular momentum with magnitudes uniformly drawn from a uniform distribution on $[-0.99, 0.99]$.

In the third dataset, noise is generated in a similar manner to the second dataset. However, a new PSD is chosen (from the same set) for each segment. The distribution of component masses is the same as in the second dataset. In contrast, the spins are no longer aligned; their magnitude is uniform from 0 to 0.99, and their direction is

isotropically distributed. All higher-order modes available to the `IMRPhenomXPHM` approximant are used.

In the fourth dataset, real LIGO noise is used. A real noise file in the extent of approximately three months has been prepared by the MLGWSC-1 team, the data generation script randomly chooses segments from it to comprise the dataset background, and the L1 stream is time shifted with respect to H1 by a random amount in order to introduce different noise realizations. The injections are generated in a manner identical to the third dataset.

B. Evaluation procedure

The evaluation is done in a similar manner to [21,22]. The submitted algorithms are applied to background data without any injections as well as to data with BBH injections to determine the relationship between their false-alarm rate (FAR) and sensitive distance.

Each submitted algorithm is required to take a file in the format described in Sec. II A as input and produce a file containing identified candidate events as output. It must be an HDF5 file containing three datasets referring to the GPS time of the events, the ranking statistics, and the tolerance for error in the time.

The evaluation is performed by the `evaluate.py` script supplied by the MLGWSC-1 [25]. It requires the outputs of the submission algorithm on both the foreground and the background files as input, identifies true positives, and determines the FAR at varying detection thresholds. The relationship between the FAR and the sensitive distance is in principle similar to the receiver operating characteristic, which is the relationship between the percentage of false positives and true positives as one varies the threshold for identification of a positive.

To obtain the sensitivity curve of an algorithm based on the identified background and foreground events, we first count the number of background events with a ranking statistic greater than the threshold. Dividing by the total duration of the background data analyzed (in this case 30 days $= 2\,592\,000$ s), we find the FAR. The sensitive volume of the search at FAR $= \mathcal{F}$ can be calculated by [33]

$$V(\mathcal{F}) = \iint \epsilon(\mathcal{F}; \mathbf{x}, \mathbf{\Lambda}) \phi(\mathbf{x}, \mathbf{\Lambda})\, \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{\Lambda}, \tag{2}$$

where $\mathbf{x}$ are an injection's spatial coordinates, $\mathbf{\Lambda}$ the other injection parameters, $\epsilon(\mathcal{F}; \mathbf{x}, \mathbf{\Lambda})$ is the efficiency of the search, and $\phi(\mathbf{x}, \mathbf{\Lambda})$ is the injection parameter distribution. If we denote $N_{I,\mathcal{F}}$ the number of found injections at a FAR $= \mathcal{F}$ and $\mathcal{M}_{c,i}$, $i = 1, \ldots, N_{I,\mathcal{F}}$ the chirp masses of the found injections, the expression simplifies to [24]

$$V(\mathcal{F}) \approx \frac{V(d_{\max})}{N_I} \sum_{i=1}^{N_{I,\mathcal{F}}} \left(\frac{\mathcal{M}_{c,i}}{\mathcal{M}_{c,\max}} \right)^{5/2}, \tag{3}$$

where N_I is the total number of injections and $\mathcal{M}_{c,\max}$ is the upper limit of injected chirp masses.

We then call the graph of the sensitive volume $V(\mathcal{F})$ as a function of the FAR the algorithm's *sensitivity curve*, and these are the main criterion for the challenge evaluation.

The runtimes of submitted algorithms were also measured and are available in the challenge paper [24]. All submitted algorithms are evaluated on standardized hardware, provided by the challenge organizers. The hardware consists of a total of eight Intel Xeon Silver 4215 cores at 2.5 GHz, 192 GB of RAM, and eight nVidia RTX 2070 GPUs with CUDA support, 8 GB of VRAM each.

III. EXPERIMENTAL SETUP

A. Data processing

As described in [11], the standard noise model in LIGO detectors is correlated in the time domain. However, using the Fourier transform, in the frequency domain the noise is uncorrelated and described by a Gaussian distribution with zero mean and a frequency-dependent variance called the PSD and denoted $S_n(f)$.

Let us use $\tilde{\mathbf{d}}$ to denote the Fourier transform of a time series $\mathbf{d}$. Following [11], the transformation

$$\mathbf{d} \mapsto \mathbf{d}_w, \qquad \tilde{d}_w(f) = \frac{\tilde{d}(f)}{\sqrt{S_n(f)}} \tag{4}$$

yields a time series with a flat PSD, corresponding to white noise. This process is called *whitening* and is a common method in GW data analysis. The PSDs in GW detectors rise steeply toward both low and high frequencies, and the signals are dominated by strong noise at frequencies outside the most sensitive band of the detectors.

Following [21], we feed whitened data to the ML model. When applying to test data, the algorithm first estimates the PSD of the time series in question using Welch's method [34] with a segment duration of 0.5 s, then symmetrically truncates the time-domain response of the $S_n(f)^{-1/2}$ whitening filter to a width of 0.25 s, and uses this PSD to whiten the entire time series. This is done for each segment in the input data separately, as well as for each detector.

As the noise in LIGO detectors is not stationary over timescales on the order of days, one must account for the PSD drift. This is addressed by slicing the data into chunks shorter than the PSD-drift timescale in the test data generation process [25].

B. Training and validation data

In the training and validation datasets, the noise is taken from the real noise file provided by the MLGWSC-1. A segment from the file is chosen at random, its PSD is estimated and used to whiten the entire segment, and the whitened segment is sliced into one-second samples. While the noise generation loop is running, these slices are used

TABLE I. Distributions from which waveform injection parameters are drawn. Intervals refer to a uniform distribution.

Parameter	Uniform distribution
Approximant	`IMRPhenomXPHM`
Component masses	$m_1 \geq m_2 \in [10M_\odot, 50M_\odot]$
Spin magnitudes	$\lvert\chi_1\rvert, \lvert\chi_2\rvert \in [0, 0.99]$
Spin directions	isotropic
Coalescence phase	$\Phi_0 \in [0, 2\pi)$
Inclination angle	$\iota \in [0, 2\pi]$
Declination	$\sin\theta \in [-1, 1]$
Right ascension	$\varphi \in [-\pi, \pi)$
Polarization angle	$\Psi \in [0, 2\pi)$
Sampling rate	2048 Hz
Low frequency cutoff	20 Hz

sequentially, and once all have been used, a new segment is whitened and sliced in the same manner. The PSD is retained through the processing of the entire segment for whitening of waveform injections.

To generate the waveform injections, we apply the Python package PyCBC [35]. The distributions of individual parameters are summarized in Table I; they follow the distribution used in test datasets 3 and 4 (see Sec. II A) with exceptions, which we describe in the following paragraphs. A limited number of noise samples (given for each experiment in Sec. IV) are injected with a waveform and assigned the label (1,0); the remaining ones remain pure noise and are assigned the label (0,1). However, the waveforms are normalized to a network optimal signal-to-noise ratio (SNR) $\rho_{\text{net}} = 1$ during the data generation procedure and only injected at a randomly generated $\rho_{\text{net}} \in [7, 20]$ at each training epoch. Owing to the SNR normalization, the luminosity distance is irrelevant, and a fiducial 1 Mpc value (the PyCBC default) is passed to the approximant.

For consistency with the experiments of [21], we set the lower mass limit to $10M_\odot$ instead of $7M_\odot$ used to generate test datasets 2–4. An additional training run confirms that including the range $[7M_\odot, 10M_\odot]$ in the training data does not improve the performance of the search. We suspect this is due to the increased length of waveforms in this region of the parameter space [24], due to which a part of the waveform's SNR is outside the network's input window when the merger is aligned.

Furthermore, owing to an oversight on our part, the inclination angle does not follow the astrophysical distribution $\cos\iota \in [-1, 1]$. However, this is not expected to pose an issue, as the dominant effect of the inclination angle on the waveforms is a constant rescaling [36], which is lost as we normalize the waveforms to a fixed network SNR. A rerun of the code for the MLGWSC-1 submission with the astrophysical distribution confirms that the results are indistinguishable.

Both the training and validation data are generated by following the steps below:

(1) Get noise:
   (a) get next slice from current segment,
   (b) if segment finished, choose a new one at random, whiten it, slice it, and take its first slice.
(2) If applicable, generate waveform:
   (a) set up parameters (see Table I),
   (b) generate waveform,
   (c) crop so that merger is within the given interval; append zeros,
   (d) whiten using the PSD of the corresponding noise segment,
   (e) normalize to optimal $\rho_{\text{net}} = 1$.
(3) Store noise and waveform separately. At each training epoch, inject at a newly generated optimal SNR.

C. Test data

Test data meant for evaluation before submitting are generated using the program `generate_data.py` supplied by the MLGWSC-1 [25]. For final testing, all four datasets are generated with the length of 2 592 000 seconds = 30 days, and the seed is set to 4 261 537. Dataset 4 with this seed is used to generate Fig. 2 and to optimize the updated submission in Sec. IV C.

The seeds used for generating the challenge datasets to evaluate the submitted algorithms to the MLGWSC-1 and to plot Fig. 4 are given in Sec. II A.

D. Machine learning

The MLGWSC-1 is aimed at evaluating the performance of ML algorithms. In its simplest form, this corresponds to a model with an arbitrary number of free parameters whose error is being optimized over a large dataset. This is frequently done using gradient-descent-based optimizers and their stochastic varieties, which approximate the gradients on small batches of the dataset in their successive iterations. The error function being optimized here is a modification of the binary cross entropy loss [21]

$$\mathcal{C}(\bar{\mathbf{Y}}, \mathbf{Y}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} Y_{ij}\log\left((1-\varepsilon)\bar{Y}_{ij} + \varepsilon\right), \quad (5)$$

designed to remove divergences when an element of $\bar{\mathbf{Y}}$ is zero using the regularization parameter $0 < \varepsilon \ll 1$.

Neural networks are a class of ML models built of artificial neurons; these are functions defined as

$$f\colon \mathbb{R}^n \to \mathbb{R}, \quad (6a)$$

$$\mathbf{x} \mapsto \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right). \quad (6b)$$

The parameters w_i and b are called the weights and bias, respectively, and are optimized through the training

process. The function σ is called an *activation function*; a popular choice we use here is the Exponential Linear Unit [37] with $\alpha = 1$:

$$\mathrm{ELU}(z) = \begin{cases} \alpha(\exp(z) - 1) & \text{if } z < 0, \\ z & \text{if } z \geq 0. \end{cases} \tag{7}$$

A feed-forward neural network is organized in layers of independent neurons, each of which feeds its output into neurons of the following layer. They can be fully connected, i.e., the input of each neuron consists of the outputs of all neurons in the previous layer, also called dense layers. In this paper, we also make use of convolutional layers, whose structure corresponds to a set of filters sliding over a multichannel input [38]. This reduces the number of independent connections and thus weights in the network.

Further components are max pooling layers, which act as a downsampling operation [39], and dropout layers, which improve the training convergence through a type of noise injection [40,41]. For an introduction to ML and neural networks, we refer the reader to [42,43].

E. Model architecture

In this work, we use a simple convolutional neural network (CNN) design, which is an extension of the architecture used in [21]. For a simple implementation of the method used there, we first define a CNN called the *base network*, which does not have a final activation. Its architecture is shown in Table II.

Unlike coincident searches such as the CNN-Coinc submission [22] to the MLGWSC-1, wherein the streams from each detector are analyzed separately and combined using a probability-based formula, we employ a coherent approach. The network accepts a two-channel input to carry data from two detector streams.

The base network produces two outputs, which we denote x_0, x_1. Following the method of [21], for training we append a Softmax layer,

$$y_i = \mathrm{Softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}, \tag{8}$$

which maps its inputs to a set of positive numbers which sum up to 1. The purpose of this activation is to represent uncalibrated probabilities [44] of different classes in classification problems, and in this case we wish the output y_0 to represent the probability that the input sample contains an astrophysical GW signal.

The networks are trained using the stochastic Adam optimizer [45] with a learning rate of $\gamma = 4 \times 10^{-6}$, and the other parameters set to their defaults in PyTorch [46] ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$), for a total of 250 epochs. The training dataset is split into batches of 32 samples, and each epoch consists of one optimizer step per batch.

TABLE II. Architecture of the base network. It accepts an input with two channels corresponding to two detector streams and possesses 635 318 trainable weights. "KS" refers to kernel size, and "shape" is the output shape of the corresponding layer. The batch normalization layer is only used in the original submission to the MLGWSC-1 but not in the improved searches.

Layer	KS	Shape	Activation
Input		2×2048	
(Batch norm)		2×2048	
Convolution	33	16×2016	ELU
Convolution	32	16×1985	ELU
Convolution	17	16×1969	ELU
Convolution	16	16×1954	ELU
Max pooling	4	16×488	
Convolution	17	16×472	ELU
Convolution	16	32×457	ELU
Convolution	9	32×449	ELU
Convolution	8	32×442	ELU
Max pooling	3	32×147	
Convolution	9	32×139	ELU
Convolution	8	64×132	ELU
Convolution	9	64×124	ELU
Convolution	8	64×117	ELU
Max pooling	2	64×58	
Flatten		3712	
Dense		128	ELU
Dropout		128	
Dense		128	ELU
Dropout		128	
Dense		2	

When testing in the same manner, however, a numerical issue arises. In single-precision floating point arithmetic using PyTorch, y_0, which we would like to use as the ranking statistic of the resulting search, rounds up to 1 when $x_0 - x_1 \gtrsim 16$, which is well in the range of values encountered by the search. To resolve this, we rewrite Eq. (8) for $i = 0$ as

$$y_0 = \frac{1}{1 + \exp(x_1 - x_0)} = \frac{1}{1 + \exp(-\Delta x)}. \tag{9}$$

We see that y_0 is a purely growing function of $\Delta x = x_0 - x_1$, which is therefore an equivalent ranking statistic, without suffering from the same numerical issue. Therefore, Δx is used as the ranking statistic in the search. This technique is called the unbounded softmax replacement. For more detailed information see Ref. [21].

The CNN is only part of the detection algorithm following [21], as it only accepts simple one-second-long slices. The full algorithm consists of feeding overlapping slices of the test data to the network, applying a threshold, and clustering the results into candidate detections. First, the entire segment is whitened using the method described in Sec. III A.

Then, the segment is sliced into one-second long samples with an offset of 0.1 seconds, which are fed to the network

and the Δx outputs recorded. Because the networks are trained on injections with merger time 0.6 to 0.8 seconds after the sample start time, each slice is associated with the time 0.6 seconds after the start time of the slice, in order to compensate for this alignment.

A threshold is applied to the network outputs, and those which exceed it are clustered by time, with a minimal separation of 0.35 seconds between clusters. Each of these clusters is then considered a candidate event to be saved in the output file. As the ranking statistic, the maximum of the network outputs in the cluster is used, and the time corresponding to the maximum is used as the time of the candidate event. For all output events, the value of 0.2 seconds is chosen as the time uncertainty in the search output (see Sec. II B), to match the size of the merger alignment interval in the training data.

IV. RESULTS

A. MLGWSC-1 submission

For the submission, we choose the training dataset to contain 500 000 pure noise samples and 500 000 noise + waveform samples, the validation dataset to contain 100 000 pure noise samples and 100 000 noise + waveform samples, and the sliced real noise is used. The training and validation losses are monitored during the training, and their evolution is shown in Fig. 1.

Out of the local minima of the validation loss, the global minimum as well as two earlier local minima are chosen and further tested by applying to the test datasets 3 and 4; the result for dataset 4 is shown in Fig. 2. The results on dataset 3 were virtually indistinguishable; for better performance on dataset 4 we chose the network state at epoch 79 for the submission to the MLGWSC-1.

It is necessary to set one more parameter: the first detection threshold, applied before clustering. Applying the trained algorithm to a shorter dataset (length of one day), the resulting sensitivities are displayed in Fig. 3. We choose the value of -8 for the threshold applied to the Δx ranking statistic, as its performance is indistinguishable from others at lower FARs, while it reaches up to higher

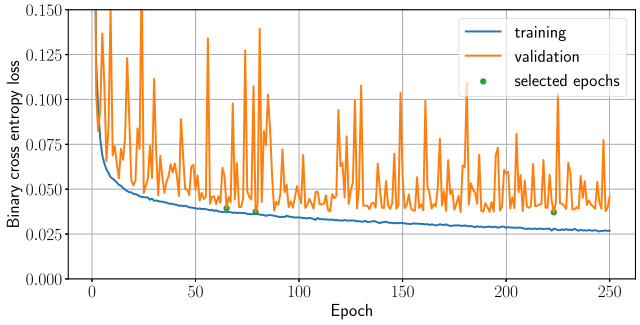

FIG. 1. Evolution of the training and validation loss values throughout the training of the MLGWSC-1 submission.

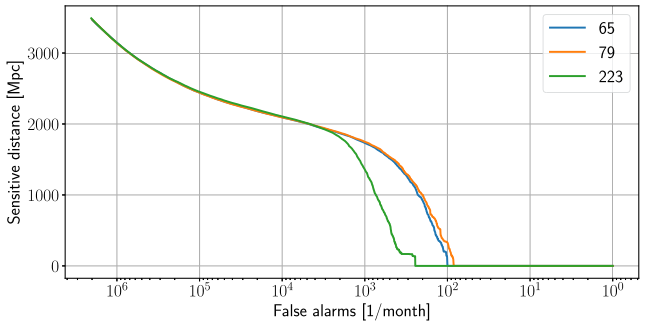

FIG. 2. Sensitivity curves of the network at three minima of the validation loss highlighted in Fig. 1 used to select the final network state for the submission.

FARs than others. However, at FAR values relevant to GW astronomy, all algorithms perform comparably; therefore this choice does not seem to be relevant, and we do not perform the same optimization in further experiments.

B. MLGWSC-1 results

The MLGWSC-1 received a total of six contributions, four of which are ML based. The remaining two are conventional analyses to provide a baseline; the first is the matched-filtering-based PyCBC [33], the other is the loosely modeled search cWB [5,47].

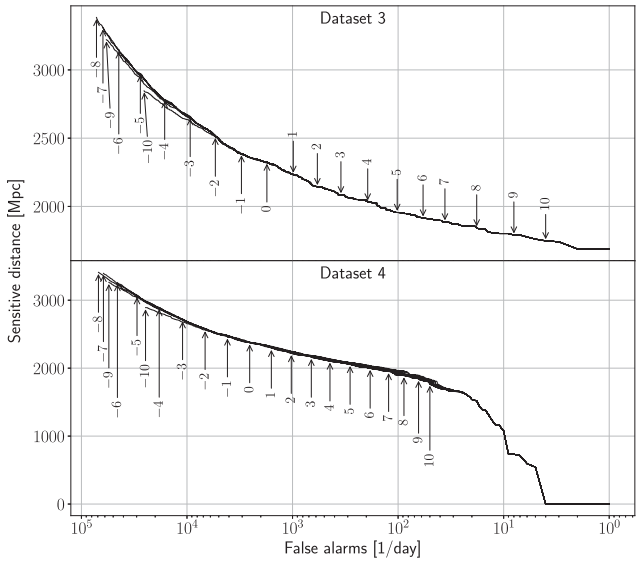

FIG. 3. Sensitivity curves of the submitted trained network using multiple Δx thresholds to determine a suitable value. The datasets are generated by the script provided by the MLGWSC-1 in the length of one day (86 400 seconds), with the difficulty specified as datasets 3 and 4 in the top and bottom panel, respectively. Owing to a large overlap between the sensitivity curves, rather than color coding, their left ends are annotated with the threshold value. All curves reach the same point at the right end, $\mathcal{F} = 1\ \text{day}^{-1}$.

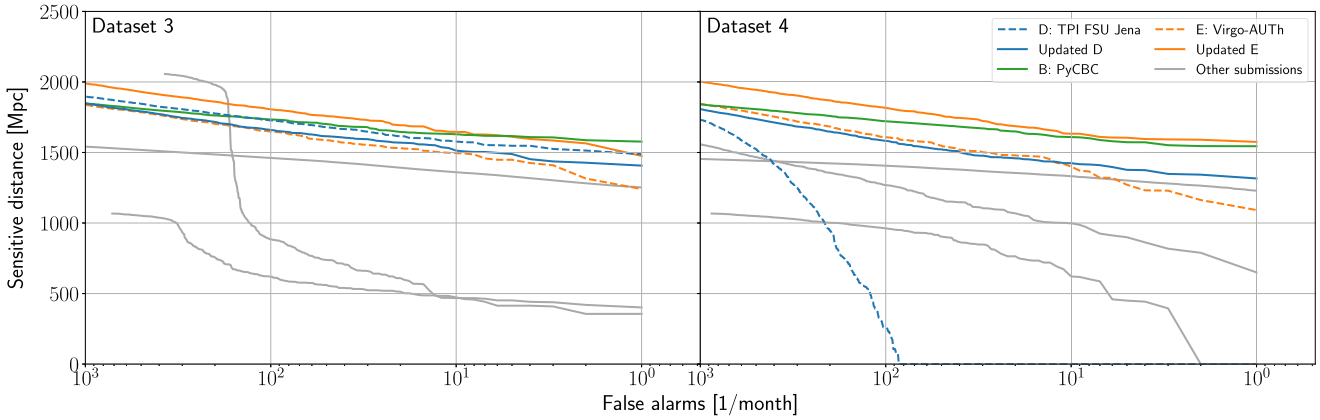

FIG. 4. Sensitivity curves of three selected submissions, along with updated versions of two of them, on datasets 3 and 4 of the MLGWSC-1. Each panel contains the performance of the submissions on one test dataset. Dashed lines mark conventional analyses, and solid lines mark ML-based search algorithms. In case of the TPI FSU Jena and Virgo-AUTh teams, the dotted lines mark the original submissions, while the solid lines mark the updated algorithms. The remaining submissions are shown in gray for illustration of overall challenge results.

Rather than an extensive coverage of the MLGWSC-1 results, which are described in great detail in [24], this section focuses on a particular issue which occurs when real noise is presented to our algorithm. We would like to specifically bring to the reader's attention the performance of our algorithm (labeled D: TPI FSU Jena) and the algorithm labeled E: Virgo-AUTh, whose sensitivity curves on datasets 3 and 4 are shown in Fig. 4. Both ML submissions are plotted as dashed lines; in addition the PyCBC submission is shown.

Both submissions use a very similar approach. In the final evaluation on test dataset 3, their performances are close to each other with D operating at a slightly higher sensitivity at all FARs; this gap widens as we approach $\mathcal{F} = 1$ month^{-1} (in fact, this holds on all datasets which use Gaussian noise [24]). However, the Virgo-AUTh algorithm retains $\geq 90\%$ of the sensitive distance of the TPI FSU Jena search at $\mathcal{F} \geq 2$ month^{-1}, and at $\mathcal{F} = 1000$ month^{-1} this gap narrows to a separation of roughly 4%.

Moving to dataset 4, the performance of the Virgo-AUTh algorithm degrades only mildly. In contrast, the performance of our submission deteriorates much more, losing all sensitivity at $\mathcal{F} < 10^2$ month^{-1}. This is due to the noise transients which are omnipresent in real detector data, and which are revealed to produce triggers louder than injected waveforms in further analysis.

Finally, the runtimes of our algorithm are consistently lower than those of the other submissions. On average, the Virgo-AUTh search takes ∼50% longer to run on the challenge hardware on all four test datasets due to the higher complexity of its network architecture. On datasets 2–4 the estimated runtimes of PyCBC are ∼40 times as large. We note that the given PyCBC runtimes are estimations as a different hardware setup is used to run the search.

C. Updated submission

Following the Virgo-AUTh team's algorithm through [24,48,49], we identify three main differences, which we expect to be responsible for the large difference in performance. These are input normalization, network architecture, and training dataset distribution, and we cover them in detail below.

1. Input normalization and network architecture

While our submission retains the batch normalization layer from [21], the Virgo-AUTh team has tested multiple input normalization methods, and uses a deep adaptive input normalization (DAIN) layer instead. Inspired by this idea, we attempt to replace the batch normalization layer by a DAIN, as well as simply remove the input normalization altogether. Out of these three options, only the complete removal of input normalization brings a discernible reduction in FARs.

While our submission makes use of a fairly simple and small CNN design, the Virgo-AUTh team's submission uses a larger and more complex ResNet design. We attempt to replace our network with an identical ResNet design and encounter no improvement in sensitivity, along with an increased computational cost. The same experiment with all three input normalization options mentioned above yields similar results. Therefore, we decide to retain the original CNN design and merely remove the batch normalization layer for further development.

2. Training dataset distribution

In the end, the key issue turns out to be the distribution of the training dataset. We use a 1:1 dataset in terms of the number of pure noise samples to ones with injections in our original submission. Further experiments indicate the

optimal ratio to be 1:3 as the network's performance degrades when this ratio is shifted in either direction.

Six training runs are performed using the same optimization procedure as previously. At each epoch, the network's sensitivity is evaluated on test data with real noise, and from each run the state with the highest sensitive distance at $\mathcal{F} = 1$ month^{-1} is chosen and labeled as `R<run number 1-6>/<4-digit epoch number>`. Of the resulting six states, we choose `R1/0021` for the final search algorithm as it has the highest sensitivity. Its sensitivity curves are shown in Fig. 4 alongside the curves of all submissions as well as the updated Virgo-AUTh search, called AResGW [48,49].

The sensitivity on datasets using Gaussian noise deteriorates slightly; this is to be expected as one optimizes for a different noise distribution, rejecting potential glitches in data containing none. At $\mathcal{F} = 1$ month^{-1}, the sensitive distance is reduced by 5.4%. In the overall ranking, ours remainsall submissions as well as the updated the most sensitive of all ML submissions on Gaussian noise.

On real noise, the updated submission reaches the highest sensitivity of all ML submissions at $\mathcal{F} \lesssim 10$ month^{-1} and is narrowly outperformed by Virgo-AUTh at higher FARs. At $\mathcal{F} = 1$ month^{-1}, our updated submission has a sensitivity distance of 1316 Mpc, and Virgo-AUTh operates at 87% of this value. At the same time, the updated version of their algorithm outperforms ours in both cases.

D. Application to O3b data

The O3 LIGO observing run was split by a commissioning break into two phases, O3a and O3b [27,50]. The first part is used to train the CNNs above to recognize BBH waveform injections in real LIGO noise. In this section, we apply the searches developed above to real data recorded by LIGO through the O3b phase and cross reference the output with the transients recorded in the GWTC-3 catalog [3].

To query O3b data, we require a minimum segment length of one minute and the same data quality requirements as the real noise file used in the MLGWSC-1; known injections are not removed. This leaves us with a total of 8 228 706 seconds of data in a total of 2 377 segments, amounting approximately to 95 days and 6 hours. In comparison, the full O3b observing run was 147 days and 2 hours in length.

We apply all six searches trained in Sec. IV C 2 to these data. The GWTC-3 catalog [3] consists of 35 confident detections and seven marginal ones. Events lying outside the segments of available data are excluded, leaving us with 31 confident and four marginal events to be found.

TABLE III. List of O3b events from the `GWTC-3-confident` catalog [3] and their identification by the six final searches. Events which are not recovered by the given search are marked by centerdots. Thirteen events are omitted (see Table IV). The events are grouped into three sections based on their estimated component masses (see text for details).

		$\mathcal{F}$ [month^{-1}]					
Event name	ρ_{MF}	`R1/0021`	`R2/0150`	`R3/0036`	`R4/0038`	`R5/0193`	`R6/0026`
GW200224_222234	20.0	0.0	0.9	0.0	0.0	0.0	0.0
GW200311_115853	17.8	0.0	0.9	0.6	1.3	0.0	6.0
GW200225_060421	12.5	0.0	1.6	0.0	0.0	1.9	0.3
GW191215_223052	11.2	0.0	1.9	1.3	1.3	0.0	1.3
GW200208_130117	10.8	19.2	2.2	3.5	3.1	1.6	10.1
GW200219_094415	10.7	5.0	4.7	8.8	38.7	12.6	19.5
GW200209_085452	9.6	1.3	2.8	0.9	2.8	2.2	0.3
GW191204_110529	8.8	1.6	3.1	0.0	3.1	5.0	3.1
GW200308_173609	7.1	···	···	···	···	···	···
GW191222_033537	12.5	0.0	4.1	2.5	2.5	0.3	0.3
GW200128_022011	10.6	25.5	3.1	0.0	11.7	10.4	2.2
GW191230_180458	10.4	6.6	149	19.5	98.9	36.9	5.0
GW191127_050227	9.2	38.7	2.8	4.4	18.6	3.1	6.6
GW200220_124850	8.5	215	517	956	96.1	695	375
GW191126_115259	8.3	···	···	···	···	···	···
GW200216_220804	8.1	···	189	···	···	841	···
GW191113_071753	7.9	···	634	391	···	713	647
GW200306_093714	7.8	485	407	720	···	69.3	···
GW200208_222617	7.4	38.1	6.0	19.5	55.1	159	187
GW200322_091133	6.0	810	898	···	···	···	···
GW191204_171526	17.5	3.5	8.8	4.1	4.4	7.6	6.0
GW191109_010717	17.3	0.0	1.9	0.9	0.6	1.3	0.9

TABLE IV. List of O3b events omitted from Table III. Events listed in the first column are omitted due to insufficient data quality in either detector, and events listed in the second column are omitted due to being missed completely by all searches. The exception is GW191105_143521, which is recovered at $\mathcal{F} = 658$ month^{-1} by the `R5/0193` search and missed by the others.

Data quality	Missed
GW200302_015811	GW191129_134029
GW200129_065458	GW200115_042309
GW200112_155838	GW200202_154313
GW191216_213338	GW200316_215756
	GW191105_143521
	GW191219_163120
	GW191103_012549
	GW200210_092254
	GW200220_061928

These excluded events are listed in the left column of Table IV. In addition, we confirm that none of the events contained in available segments take place closer than 46 seconds to either end of their respective segments.

A catalog event is marked as found, if the search output contains an event within 0.2 seconds of the time given in the catalog, and it is assigned its corresponding ranking statistic t. The remaining catalog events are considered missed, and the remaining events reported by the search are considered false alarms. The catalog event is then considered detected at a FAR of

$$\mathcal{F} = \frac{N_{f>t}}{T}, \tag{10}$$

where $N_{f>t}$ is the number of false alarms louder than t, and T is the total length of the analyzed segments. In addition, if the FAR of an event is at least 1000 month^{-1}, it is also considered missed.

None of the events marked marginal in the GWTC-3 catalog are found by either of the searches. The resulting FARs of confident events in the analyzed segments are shown in Table III. The table is split into three sections: in the first, the 90% credible intervals on both component masses lie fully in the $[10M_\odot, 50M_\odot]$ range used for training the networks, while in the third, at least one of them lies fully outside $[10M_\odot, 50M_\odot]$. The remaining cases are contained in the second section. The credible intervals and accompanying SNR values come from the catalog's parameter estimation pipeline based on Bilby [51,52] and are supplied by GWOSC [53].

Let us comment shortly on the results of Table III. Most importantly, all events in the first section, where the search algorithms are expected to operate at a high sensitivity, are found by all six tested networks at a FAR lower than 40 month^{-1}, with the exception of GW200308_173609, which is the second weakest event in the catalog at $\rho_{\mathrm{MF}} = 7.1$. In the vast majority, the events are detected at $\mathcal{F} < 4$ month^{-1}.

In the second and third sections the searches are expected to operate at a reduced sensitivity as the corresponding parameter space is not fully covered in the training dataset. This is confirmed in Table III; however, louder events at $\rho_{\mathrm{MF}} \gtrsim 9$ and $\rho_{\mathrm{MF}} \gtrsim 17$ in the second and third section, respectively, are also mostly detected at $\mathcal{F} < 10$ month^{-1} by the ML-based searches.

As a final comment, Q-scan spectrograms of the loudest false alarms in the analyzed data seem to be consistent with them being known types of glitches.

V. CONCLUSION

We have presented a convolutional neural network-based gravitational wave detection algorithm capable of performing comparably to conventional algorithms in specific settings, and its implementation, submitted to the MLGWSC-1. While the submission performs well on test data using Gaussian noise, the noise transients present in the data with real noise prove to be too much of a challenge and reduce its sensitivity to zero at relevant FARs. In the present work, we resolve this issue by a careful optimization of the training parameters and demonstrate that the updated search outperforms all other original challenge submissions besides the PyCBC matched-filter search.

At the same time, while each independent run of the updated algorithm converges to a state with high sensitivity of the resulting search, a detailed analysis reveals that the sensitivity is highly nonmonotonic during the training [54]. In addition, Fig. 1 also shows unexpected oscillations in the validation loss. This phenomenon is not yet fully understood and warrants further investigation.

As a final application of the updated search, we analyze open data from the O3b observing run [27] of the LIGO-Virgo Collaboration and cross reference the results with the corresponding catalog GWTC-3 [3]. We demonstrate that in the intended regime of BBHs with component masses between $10M_\odot$ and $50M_\odot$, our searches can confidently detect events with a network SNR above 8. This is in line with contemporary matched-filter-based searches, as the value 8 roughly corresponds to one false alarm per month [21].

Full outputs of all six search algorithms as well as spectrograms of the 128 loudest events of each are publicly available in the data release [55].

ACKNOWLEDGMENTS

O. Z. thanks the Carl Zeiss Foundation for the financial support within the scope of the program line "Breakthroughs" and is supported by the fellowship Lumina Quaeruntur No. LQ100032102 of the Czech Academy of Sciences. Further support has been provided by the COST network CA17137 "G2net." The computational experiments were performed on the ARA cluster at the Friedrich-Schiller-Universität Jena and the Atlas cluster

financed by the Gottfried Wilhelm Leibniz Universität Hannover and the Max-Planck-Gesellschaft through the Albert-Einstein-Institut Hannover. We thank the Observational Relativity and Cosmology division for access. F. O. acknowledges support by the Max Planck Independent Research Group Program. Special thanks go to Marlin Schäfer for his great contribution through his work on organizing the MLGWSC-1 as well as discussions, and to all contributors to the challenge. This research has made use of data or software obtained from the Gravitational Wave Open Science Center (gwosc.org), a service of the LIGO Scientific Collaboration, the Virgo Collaboration, and KAGRA. This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation, as well as the Science and Technology Facilities Council (STFC) of the United Kingdom, the Max-Planck-Society (MPS), and the State of Niedersachsen/Germany for support of the construction of Advanced LIGO and construction and operation of the GEO600 detector. Additional support for Advanced LIGO was provided by the Australian Research Council. Virgo is funded, through the European Gravitational Observatory (EGO), by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale di Fisica Nucleare (INFN), and the Dutch Nikhef, with contributions by institutions from Belgium, Germany, Greece, Hungary, Ireland, Japan, Monaco, Poland, Portugal, and Spain. KAGRA is supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan Society for the Promotion of Science (JSPS) in Japan; the National Research Foundation (NRF) and Ministry of Science and ICT (MSIT) in Korea; Academia Sinica (AS) and the National Science and Technology Council (NSTC) in Taiwan.

APPENDIX: CODE

The code provided by the MLGWSC-1 organizers is available in [25]. The code used in the experiments is contained in [55]. The submission to the MLGWSC-1 as described in Secs. III and IV A is stored in the directory `mlgwsc-1`. For the experiments detailed in Secs. IV C and IV D, the code is available in the subdirectory `correction` along with additional materials and results.

[1] B. P. Abbott, R. Abbott, T. D. Abbott, S. Abraham, F. Acernese, K. Ackley, C. Adams, R. X. Adhikari, V. B. Adya, C. Affeldt, M. Agathos, K. Agatsuma, N. Aggarwal, O. D. Aguiar, L. Aiello *et al.* (LIGO Scientific and Virgo Collaborations), Astrophys. J. Lett. **882**, L24 (2019).

[2] R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, N. Adhikari, R. X. Adhikari, V. B. Adya, C. Affeldt, D. Agarwal, M. Agathos, K. Agatsuma, N. Aggarwal, O. D. Aguiar, L. Aiello *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), Phys. Rev. X **13**, 011048 (2023).

[3] R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, N. Adhikari, R. X. Adhikari, V. B. Adya, C. Affeldt, D. Agarwal, M. Agathos, K. Agatsuma, N. Aggarwal, O. D. Aguiar, L. Aiello *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), Phys. Rev. X **13**, 041039 (2023).

[4] S. Klimenko, S. Mohanty, M. Rakhmanov, and G. Mitselmakher, Phys. Rev. D **72**, 122002 (2005).

[5] S. Klimenko, G. Vedovato, M. Drago, F. Salemi, V. Tiwari, G. A. Prodi, C. Lazzaro, K. Ackley, S. Tiwari, C. F. Da Silva, and G. Mitselmakher, Phys. Rev. D **93**, 042004 (2016).

[6] R. Lynch, S. Vitale, R. Essick, E. Katsavounidis, and F. Robinet, Phys. Rev. D **95**, 104046 (2017).

[7] A. H. Nitz, A. Lenon, and D. A. Brown, Astrophys. J. **890**, 1 (2020).

[8] I. Harry, S. Privitera, A. Bohé, and A. Buonanno, Phys. Rev. D **94**, 024012 (2016).

[9] S. Schmidt, B. Gadre, and S. Caudill, Phys. Rev. D **109**, 042005 (2024).

[10] I. Harry, J. C. Bustillo, and A. H. Nitz, Phys. Rev. D **97**, 023004 (2018).

[11] B. P. Abbott, R. Abbott, T. D. Abbott, S. Abraham, F. Acernese, K. Ackley, C. Adams, V. B. Adya, C. Affeldt, M. Agathos, K. Agatsuma, N. Aggarwal, O. D. Aguiar, L. Aiello, A. Ain *et al.* (LIGO Scientific and Virgo Collaborations), Classical Quantum Gravity **37**, 055002 (2020).

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

[13] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).

[14] A. J. K. Chua and M. Vallisneri, Phys. Rev. Lett. **124**, 041102 (2020).

[15] S. R. Green, C. Simpson, and J. Gair, Phys. Rev. D **102**, 104057 (2020).

[16] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, Nat. Phys. **18**, 112 (2021).

[17] P. Bacon, A. Trovato, and M. Bejger, Mach. Learn. **4**, 035024 (2023).

[18] H. Shen, E. A. Huerta, E. O'Shea, P. Kumar, and Z. Zhao, Mach. Learn. **3**, 015007 (2021).

[19] S. Schmidt, M. Breschi, R. Gamba, G. Pagano, P. Rettegno, G. Riemenschneider, S. Bernuzzi, A. Nagar, and W. Del Pozzo, Phys. Rev. D **103**, 043020 (2021).

[20] E. Cuoco, J. Powell, M. Cavaglià, K. Ackley, M. Bejger, C. Chatterjee, M. Coughlin, S. Coughlin, P. Easter, R. Essick, H. Gabbard, T. Gebhard, S. Ghosh, L. Haegel, A. Iess *et al.*, Mach. Learn. **2**, 011002 (2020).

[21] M. B. Schäfer, O. Zelenka, A. H. Nitz, F. Ohme, and B. Brügmann, Phys. Rev. D **105**, 043002 (2022).

[22] M. B. Schäfer and A. H. Nitz, Phys. Rev. D **105**, 043003 (2022).
[23] M. B. Schäfer, F. Ohme, and A. H. Nitz, Phys. Rev. D **102**, 063015 (2020).
[24] M. B. Schäfer, O. Zelenka, A. H. Nitz, H. Wang, S. Wu, Z.-K. Guo, Z. Cao, Z. Ren, P. Nousi, N. Stergioulas, P. Iosif, A. E. Koloniari, A. Tefas, N. Passalis, F. Salemi *et al.*, Phys. Rev. D **107**, 023021 (2023).
[25] M. Schäfer, O. Zelenka, P. Müller, and A. H. Nitz, MLGWSC-1 Release v1.4, 10.5281/zenodo.7107410 (2022).
[26] J. Aasi, B. P. Abbott, R. Abbott, T. Abbott, M. R. Abernathy, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, V. Adya, C. Affeldt, N. Aggarwal, O. D. Aguiar, A. Ain *et al.* (LIGO Scientific Collaboration), Classical Quantum Gravity **32**, 074001 (2015).
[27] R. Abbott, H. Abe, F. Acernese, K. Ackley, S. Adhicary, N. Adhikari, R. X. Adhikari, V. K. Adkins, V. B. Adya, C. Affeldt, D. Agarwal, M. Agathos, O. D. Aguiar, L. Aiello, A. Ain *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), Astrophys. J. Suppl. Ser. **267**, 29 (2023).
[28] The HDF Group, Hierarchical Data Format, version 5 (1997-2023), https://www.hdfgroup.org/HDF5/.
[29] B. Abbott, R. Abbott, R. Adhikari, J. Agresti, P. Ajith, B. Allen, R. Amin, S. B. Anderson, W. G. Anderson, M. Arain, M. Araya, H. Armandula, M. Ashley, S. Aston, P. Aufmuth *et al.* (The LIGO Scientific Collaboration), Phys. Rev. D **77**, 062002 (2008).
[30] 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).
[31] LIGO Scientific Collaboration, LIGO Algorithm Library—LALSuite, free software (GPL) (2018), https://git.ligo.org/lscsoft/lalsuite.
[32] P. Fritschel, LIGO Document No. T070247-v1, 2009, https://dcc.ligo.org/LIGO-T070247/public.
[33] S. A. Usman, A. H. Nitz, I. W. Harry, C. M. Biwer, D. A. Brown, M. Cabero, C. D. Capano, T. Dal Canton, T. Dent, S. Fairhurst, M. S. Kehl, D. Keppel, B. Krishnan, A. Lenon, A. Lundgren *et al.*, Classical Quantum Gravity **33**, 215004 (2016).
[34] P. Welch, IEEE Trans. Audio Electroacoust. **15**, 70 (1967).
[35] A. H. Nitz, I. Harry, D. Brown, C. M. Biwer, and J. Willis, gwastro/pycbc: v2.4.0 release of PyCBC, 10.5281/zenodo.10013996 (2023).
[36] S. A. Usman, J. C. Mills, and S. Fairhurst, Astrophys. J. **877**, 82 (2019).
[37] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, arXiv:1511.07289.
[38] Y. LeCun and Y. Bengio, Convolutional networks for images, speech, and time series, in *The Handbook of Brain Theory and Neural Networks* (MIT Press, Cambridge, MA, USA, 1998), pp. 255–258.
[39] A. Zafar, M. Aamir, N. Mohd Nawi, A. Arshad, S. Riaz, A. Alruban, A. K. Dutta, and S. Almotairi, Appl. Sci. **12**, 8643 (2022).
[40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, arXiv:1207.0580.
[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, J. Mach. Learn. Res. **15**, 1929 (2014), https://jmlr.org/papers/v15/srivastava14a.html.
[42] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, Phys. Rep. **810**, 1 (2019).
[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016), http://www.deeplearningbook.org.
[44] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, in *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research Vol. 70, edited by D. Precup and Y. W. Teh (PMLR, Cambridge, MA, 2017), pp. 1321–1330, arXiv:1706.04599.
[45] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.
[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison *et al.*, in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., Red Hook, NY, 2019), pp. 8024–8035.
[47] S. Klimenko, G. Vedovato, V. Necula, F. Salemi, M. Drago, R. Poulton, E. Chassande-Mottin, V. Tiwari, C. Lazzaro, B. O'Brian, M. Szczepanczyk, S. Tiwari, and V. Gayathri, cWB pipeline library: 6.4.1, 10.5281/zenodo.5798976 (2021).
[48] P. Nousi, A. E. Koloniari, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, Phys. Rev. D **108**, 024022 (2023).
[49] P. Nousi, A. E. Koloniari, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, AResGW: Augmentation and RESidual networks for Gravitational Wave detection (2022), https://github.com/vivinousi/gw-detection-deep-learning.
[50] B. P. Abbott, R. Abbott, T. D. Abbott, S. Abraham, F. Acernese, K. Ackley, C. Adams, V. B. Adya, C. Affeldt, M. Agathos, K. Agatsuma, N. Aggarwal, O. D. Aguiar, L. Aiello, A. Ain *et al.* (KAGRA, LIGO Scientific, and Virgo Collaborations), Living Rev. Relativity **23**, 3 (2020).
[51] G. Ashton, M. Hübner, P. D. Lasky, C. Talbot, K. Ackley, S. Biscoveanu, Q. Chu, A. Divakarla, P. J. Easter, B. Goncharov, F. H. Vivanco, J. Harms, M. E. Lower, G. D. Meadors, D. Melchor *et al.*, Astrophys. J. Suppl. Ser. **241**, 27 (2019).
[52] I. M. Romero-Shaw, C. Talbot, S. Biscoveanu, V. D'Emilio, G. Ashton, C. P. L. Berry, S. Coughlin, S. Galaudage, C. Hoy, M. Hübner, K. S. Phukon, M. Pitkin, M. Rizzo, N. Sarin, R. Smith *et al.*, Mon. Not. R. Astron. Soc. **499**, 3295 (2020).
[53] LIGO Scientific and Virgo Collaborations, LIGO-Virgo strain data from GWTC-3 Catalog (2021), https://www.gw-openscience.org/GWTC-3.
[54] O. Zelenka, Applications of machine learning to gravitational waves, Ph.D. thesis, Friedrich-Schiller-Universität Jena, 2023, https://www.db-thueringen.de/receive/dbt_mods_00059058.
[55] O. Zelenka, MLGWSC-1 Submission (2022), https://github.com/ondrzel/ml-gw-search.