Ranking candidate signals with machine learning in low-latency searches for gravitational waves from compact binary mergers

Kyungmin Kim,[1,2,*] Tjonnie G. F. Li,[2] Rico K. L. Lo,[2,3] Surabhi Sachdev,[3,4] and Robin S. H. Yuen[2]

[1]*Korea Astronomy and Space Science Institute, 776 Daedeokdae-ro, Yuseong-gu, Daejeon 34055, Republic of Korea*
[2]*Department of Physics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong*
[3]*LIGO Laboratory, California Institute of Technology, MS 100-36, Pasadena, California 91125, USA*
[4]*Department of Physics, Pennsylvania State University, University Park, Pennsylvania 16802, USA*

(Received 15 December 2019; accepted 11 March 2020; published 7 April 2020)

In the multimessenger astronomy era, accurate sky localization and low latency time of gravitational-wave (GW) searches are keys in triggering successful follow-up observations on the electromagnetic counterpart of GW signals. We, in this work, study the feasibility of adopting a supervised machine learning (ML) method for scoring rank on candidate GW events. We consider two popular ML methods, random forest and neural networks. We observe that the evaluation time of both methods takes tens of milliseconds for $\sim 45,000$ evaluation samples. We compare the classification efficiency between the two ML methods and a conventional low-latency search method with respect to the true positive rate at given false positive rate. The comparison shows that about 10% improved efficiency can be achieved at lower false positive rate $\sim 2 \times 10^{-5}$ with both ML methods. We also present that the search sensitivity can be enhanced by about 18% at $\sim 10^{-11}$ Hz false alarm rate. We conclude that adopting ML methods for ranking candidate GW events is a prospective approach to yield low latency and high efficiency in searches for GW signals from compact binary mergers.

DOI: 10.1103/PhysRevD.101.083006

I. INTRODUCTION

Recently, ground-based gravitational-wave (GW) observatories, LIGO [1] and Virgo [2] detected GW170817 [3] in about 1.7 seconds advance of the observation of a short GRB, GRB170817A [4] which was identified by the Fermi gamma-ray burst monitor (GBM) [5]. These coincident observations of both GW and short GRB became a monumental event for opening the era of multimessenger astronomy [6]. From the joint observation, one of the most plausible scenarios for the central engine which powers a short GRB is confirmed too.

With the opening of the multimessenger astronomy era, it is natural to believe that we will observe other kinds of joint GW-electromagnetic (EM) events too as summarized in Ref. [7] with future GW detectors and optical telescopes such as the Large Synoptic Survey Telescope [8]. For a joint GW-EM observation, we may use a GW event as a precursor for triggering follow-up observations on its EM counterpart. The success of this kind of joint observation will strongly depend not only on reducing the error of sky localization in GW detection but also on curtailing the latency of GW search; precise sky localization is related to how many GW detectors in various geographical locations are on-line simultaneously while the latency of search is associated with the quality of GW data and analysis efficiency. Here, the efficiency implies the accuracy of analysis. However, increasing the number of GW detectors is not trivial despite KAGRA [9] and LIGO-India [10] will come on-line in the near future in addition to currently operating LIGO and Virgo detectors. Improving the quality of GW data faces another difficulty because the current instrumental specifications are adopting state-of-the-art technology already. On the other hand, enhancing the efficiency of data analysis is relatively capable since studying the capability of a new method is much easier than others. Therefore, we focus on the analysis efficiency in this work.

Up to date, several pipelines [11–16] for the low-latency GW search have been developed and conducted to search GW signals by analyzing the time series GW data in real time. The common goal of these pipelines is identifying a candidate GW event as soon as possible. Currently, the latency between the actual event time and the identification of a candidate event with those search pipelines takes about a few minutes as reported in Ref. [6] for the detection of GW170817. When a low-latency search pipeline succeeds in the identification of a candidate event based on the significance of a ranking method of each pipeline and obtains the information, e.g., event time, sky location (right

[*]kkim@kasi.re.kr

ascension and declination), and signal-to-noise ratio of the candidate event, it forwards the information to a database system, GraceDB [17]. Then GraceDB delivers the information to EM partner observatories/telescopes through an alert system such as Gamma-ray Coordinates Network [18] alert to trigger follow-up observations for seeking correlated EM events.

Meanwhile, a candidate GW event is a survived one from multiple stages of sanity tests of a search pipeline and the event contains the result of each sanity test too in addition to the observational information forwarded to GraceDB. Thus, we can regard identifying the origin of a candidate event as a *multivariate classification problem* and the information describing candidate GW events seamlessly leads to the consideration of machine learning (ML). Indeed ML has been gradually implemented and accepted in various GW data analyses [19–29] to achieve efficient, that is, accurate analysis not only for the identification of GW signals but also for the characterization of non-Gaussian transient noises. From these studies it has been shown that we can consider ML as an alternative and complementary method to the conventional ranking method of each analysis based on their classification performances.

In specific, recent studies [27,28] suggest that introducing deep learning (DL), a subset of ML, for searching transient GW signals from compact binary mergers can be considered as a breakthrough approach in achieving the enhancement of the analysis time with maintaining comparable search performance to the conventional search method. However, as discussed by the authors of Ref. [29], DL-based search methods may meet a concern whether it is suitable tool to claim statistically significant detections or not. Hence, in order to minimize such a concern, we study the feasibility of enhancing detection sensitivity of a conventional low-latency search method by adopting ML for scoring rank on candidate events, the output of the conventional method, based on the statistical and physical parameters of the candidate events.

This paper is organized as follows: we present brief descriptions on used tools, data preparation, and procedure of applying MLs in Sec. II. The result of classification performance of ML for the given data is summarized in Sec. III. In Sec. IV, we present the detection sensitivity obtained with the application of ML and compare it to the sensitivity obtained with the ranking method of a conventional low-latency search pipeline. Finally, in Sec. V, we discuss the results of this work.

II. METHOD

In this section, we summarize the methods used for configuring the input data and conducting classification of candidate events with considered MLs. In specific, for the configuration of input data, we select six statistical and physical feature parameters from the output of a low-latency search pipeline, GstLAL inspiral search pipeline [11] (hereafter GstLAL pipeline for convenience). We start with briefly introducing each tool used in Sec. II A.[1] Then we describe the procedures from preparing input data to obtaining the output of ML through Secs. II B and II C.

A. Tools

1. GstLAL inspiral search pipeline

The GstLAL pipeline is designed for the low-latency search for gravitational waves (GWs) radiated from compact binary mergers. It is built based on the GstLAL library [30] which was derived from the GStreamer [31] and the LIGO Algorithm Library [32]. The pipeline produces candidate events from data of each GW detector by performing *matched filtering* [33] with template waveforms. In turn, if two or more detectors are on-line, the pipeline searches coincident events from detectors in the network; given an event in one detector, the pipeline checks for corresponding events in the other detector within a relevant time window, which takes into account the maximum GW travel time between detectors and statistical uncertainty in the measured event time due to detector noise at the moment [11].

We can use the pipeline in two different modes, *on-line* which makes low-latency identification of a candidate event and *off-line* which archives GW data with other information such as background statistics and data quality for further investigation on the candidate event identified from the *on-line* mode. With the off-line mode, specifically, it is possible to perform the *software injection*—injecting a bunch of simulated GW signals for compact binary systems into the calibrated GW data in order to test the search performance of the pipeline by comparing the physical parameters for the simulated signals and the recovered parameters by the pipeline.

Both modes of GstLAL pipeline use *log-likelihood ratio* [34] defined as

$$\ln L(\boldsymbol{\lambda}) = \ln \frac{P(\boldsymbol{\lambda}|s)}{P(\boldsymbol{\lambda}|n)} \tag{1}$$

as a ranking method to judge the significance of candidate events. In Eq. (1), $P(\boldsymbol{\lambda}|s)$ and $P(\boldsymbol{\lambda}|n)$ are the probability of observing parameters of $\boldsymbol{\lambda}$ of candidate events of all detectors given a GW signal, s, and background noise, n, respectively. The parameter vector $\boldsymbol{\lambda}$ consists of characteristic parameters of the candidate event such as signal-to-ratio, χ^2, physical parameters of template waveforms used in identifying candidate events, detector sensitivities at the time of the event, mean trigger rates at the time of the

[1]Since describing details of used tools are out of the scope of this work, we recommend the reader to refer to the references.

event, the trigger phases, and interdetector time differences (for details, see Refs. [34,35]).

2. Machine learning

We consider *supervised* machine learning algorithms in this work since we will use *prelabeled*[2] data for training as described in the following subsection. Amongst many supervised learning algorithms, we adopt *random forest* (RF) [36] and *neural network* (NN) [37].

RF was suggested to remedy the biased overestimation problem of the classical *decision tree* algorithm. RF is basically a collection of decision trees. However, it can reduce any biased effect in data classification by imposing (i) random selection in configuring input data for each tree and (ii) random choice on criteria at each binary split. Also, as an additional method for reducing the biased overestimation problem, RF scores ranks on samples in the input data by averaging those ranks obtained from all decision trees in the forest.

NN operates as an artificial intelligence network similar to the biological neuron system: activation of a node is determined by an activation function which judges whether the strength—sum of the value of each node times the value of connection from one node to another node in the next adjacent layer—exceeds a certain criteria or not. Nowadays, NN can be divided into two categories, *shallow* NN (SNN) and *deep* NN (DNN), by the complexity of the structure of a network, more precisely, by the number of hidden layers: if there is one hidden layer, it is called SNN and if there are two or more hidden layers, it is called DNN. But it is known that if one can solve the issue on the computing time due to the complex structure of DNN, the performance of DNN is better than that of SNN in general.

In the implementation of these two MLs, we use two different open source packages: for RF, we use Scikit-Learn [38] which supports various purposes of implementing machine learning algorithms such as supervised/unsupervised learning or classification/regression problems. On the other hand, for NN, we use TensorFlow [39] because it allows constructing DNN efficiently by reducing computing time with sophisticated computational algorithms and/or multiple computing processors.

B. Data preparation

We use data obtained from LIGO Hanford, WA, USA (H1) and LIGO Livingston, LA, USA (L1). For the purpose of conducting classification with supervised MLs, we consider two classes of data, simulated signal data and background noise data. Hereafter we call the simulated signal data and background noise data simply signal data and noise data for convenience. In particular, for the signal data, we consider binaries of black hole-black hole (BBH) because GWs from BBHs are the most common type of signal detectable by ground-based detectors.[3] Hence, we use simulated data[4] used for the estimation of event rate of O1 BBH events including GW150914 [46–48]. The simulated data is obtained by the software injection with SEOBNRv2 waveform model [49] via the off-line mode of GstLAL pipeline. Physical parameters of the software injection are given in Table I. For the noise data, we use the data obtained by running a so-called *time slide* [50], which estimates the background noise by performing time shifting to outputs of GstLAL pipeline obtained from different detectors in a network, around the time of software injection. The software injection and time slide were conducted with a chosen data segment taken between October 21, 2015 UTC and December 3, 2015 UTC, where no GW signal was found during the O1 operation of LIGO and Virgo.

TABLE I. Physical parameters of software injection.

Parameter	Range/Condition
Component mass	$m_1, m_2 \geq 2\ M_\odot$ $m_1 + m_2 \leq 100\ M_\odot$ $m_1 > m_2$
Distance	[3, 5000] Mpc
z-component of spin	$s_1z, s_2z \in [-1, 1]$

From the coincident events of H1 and L1 data, we extract six feature parameters; signal-to-noise ratio (SNR) and chi-square statistic of each trigger as statistical feature parameters and, as physical feature parameters, masses and spin magnitudes of two component compact objects. We use the same feature parameters for the configuration of both signal and background data consistently.

We first shuffle the samples of the input data to reduce any biased effect in the composition of samples. Then, we divide the shuffled input data into two categories, train and test data, such as 75% of the whole data for the train data and the rest 25% for the test data. We use the train data and test data to train ML and to evaluate the performance of trained ML, respectively. The number of signal and noise samples for train and test data is tabulated in Table II. One can recognize that the number of signal and noise samples are imbalanced which may lead to biased training.

[2]If one has *unlabeled* data and wants to train a machine learning algorithm for either classification or regression, this kind of training is called *unsupervised* learning.

[3]In general, expecting electromagnetic (EM) counterparts for BBH events is mainly discussed in theoretical studies [40–44]. However a possible association of a gamma-ray burst to the first GW detection, GW150914 was discussed in the literature [45]. Thus considering BBH signals in this work is viable about discussing latency with keeping in mind GW-EM joint observations.

[4]To avoid overfitting that could be occurring in the training of MLs, we need a sufficient number of signal samples, at least $> \mathcal{O}(10^3)$. For this reason, simulated data is more preferable than a single real signal of GW150914.

TABLE II. Number of signal and noise samples for training and test data.

		Signal	Noise
H1	Train	3,641	129,405
	Test	1,220	43,129
L1	Train	3,623	129,423
	Test	1,238	43,111

However, the imbalance may mimic the real situation since, in real detections, identifying a GW signal from noise dominant GW data is common. Thus we admit the imbalance and aim that successfully classifying desired signal samples from a much larger number of noise samples is a challenge of this work.

C. Training and evaluation

We train each ML in a different manner not only because of the different characteristics of tested MLs, RF and NN, but because of the different properties and usages of implemented packages, Scikit-Learn and TensorFlow. For given data, optimal choice on the hyperparameters of MLs in the training procedure is critically related to the performance of each ML. We determine the hyperparameters of RF and NN with the strategies described in Appendices A and B, respectively, and use them to train each ML. Once the training is done, the trained ML is recalled for the evaluation of test data.

We evaluate the test data by using the trained MLs. At this stage, each ML scores a rank, r, on each sample of the test data based on the probabilistic prediction. Thus, the value of r is given within a range of $0 \leq r \leq 1$. We observe that the evaluation time for scoring ranks on about 45,000 samples in the test data takes about tens of milliseconds.

III. CLASSIFICATION PERFORMANCE

We discuss the classification performance of trained MLs in this section by comparing it to the performance of the ranking method of GstLAL pipeline.

As described in the previous section, MLs return only probabilistic values between 0 and 1 while GstLAL returns unnormalized values of the log-likelihood ratio with Eq. (1). To address this issue, we compute the log-likelihood ratio with the resulted ranks obtained from the evaluation such as

$$\ln L(r) = \ln \frac{P(r|s)}{P(r|n)} \tag{2}$$

by following the same analogy of Eq. (1) because (i) we can separate samples of the test data into either s or n based on the prelabeled class and (ii) we can estimate the probability density function of the given ranks of signal and noise samples too. Thus, by these two reasons, this approach is applicable to the evaluated result too and this prescription makes the comparison fair.

The estimation of probability densities of the numerator and denominator of Eq. (2) is done by using the *kernel density estimation* method of Scikit-Learn with Gaussian kernel and an empirically determined optimal bandwidth of 0.03. One can find the result of probability density estimation from Appendix C.

A. ROC curve

In order to discuss the performance, we draw the receiver operating characteristic (ROC) curve as a figure of merit. To draw the ROC curve, we define *true positive rate* (TPR) and *false positive rate* (FPR) as follows:

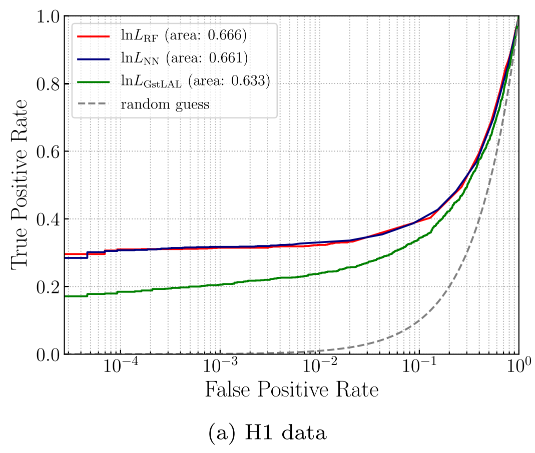
$$\begin{aligned} \mathrm{TPR} &\equiv \frac{N^{(s)}(\ln L^{(s)}(\ln L \geq \ln L_{\mathrm{th}}))}{N_{\mathrm{T}}^{(s)}} \\ &\equiv P(\ln L^{(s)}(\ln L \geq \ln L_{\mathrm{th}})), \end{aligned} \tag{3}$$

$$\begin{aligned} \mathrm{FPR} &\equiv \frac{N^{(n)}(\ln L^{(n)}(\ln L \geq \ln L_{\mathrm{th}}))}{N_{\mathrm{T}}^{(n)}} \\ &\equiv P(\ln L^{(n)}(\ln L \geq \ln L_{\mathrm{th}})), \end{aligned} \tag{4}$$

where $N^{(s)}$ and $N^{(n)}$ respectively denote the number of signal and noise samples satisfying their values of $\ln L$ larger than or equal to a given threshold value, $\ln L_{\mathrm{th}}$ within the group of signal samples, $\ln L^{(s)} = \{\ln L_i; i = 1, 2, \ldots, N_{\mathrm{T}}^{(s)}\}$ and the group of noise samples, $\ln L^{(n)} = \{\ln L_j; j = 1, 2, \ldots, N_{\mathrm{T}}^{(n)}\}$. Therefore, $\ln L^{(s)}(\ln L \geq \ln L_{\mathrm{th}})$ or $\ln L^{(n)}(\ln L \geq \ln L_{\mathrm{th}})$ represents subgroups of $\ln L^{(s)}$ or $\ln L^{(n)}$, respectively, satisfying $\ln L \geq \ln L_{\mathrm{th}}$. $N_{\mathrm{T}}^{(s)}$ and $N_{\mathrm{T}}^{(n)}$ are respectively the total number of signal and noise samples of test data presented in Table II. Note that TPR and FPR represents how likely it is to identify signal sample as signal correctly and noise sample as signal incorrectly respectively. Hence, we desire to obtain a higher value of TPR than FPR as $\ln L_{\mathrm{th}}$ increases. Subsequently, we can interpret a ranking method resulting higher in TPR at lower FPR as a better discriminator in distinguishing signal from noise adequately.

We present ROC curves in Fig. 1 by computing TPRs and FPRs with Eqs. (3) and (4) respectively. To depict the tendency of TPR with respect to FPR, we limit the range of $\ln L_{\mathrm{th}}$ as $\ln L_{\mathrm{min}}^{(n)} \leq \ln L_{\mathrm{th}} \leq \ln L_{\mathrm{max}}^{(n)}$, i.e., to make the minimum FPR to be $1/N_{\mathrm{T}}^{(n)}$. In the legend box of Fig. 1, we also present the *area under curve* (AUC) for each result because it represents the probability that a ranking method will score a higher value on an arbitrary signal instance than the value of an arbitrary noise instance. For the computation of AUC, we use the *trapezoidal* method.

From this figure, we can see that all cases are drawn in the upper region of the gray-dashed line which indicates



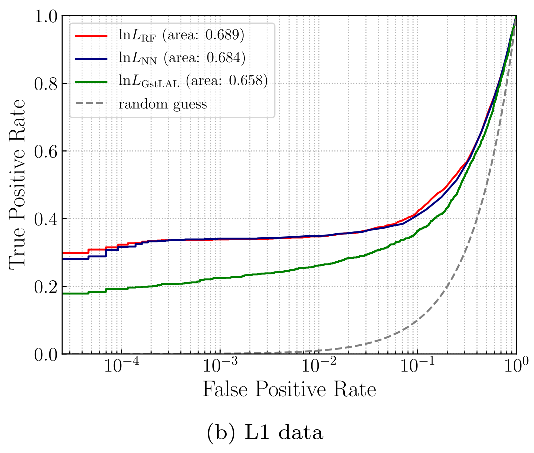


FIG. 1. Combined ROC curves and values of area under curve. The left and right panels show the ROC curves of the result of H1 or L1 data, respectively. The red and navy solid lines indicate the results of $\ln L$s of MLs and the green solid line indicates the result of $\ln L$ of GstLAL pipeline. One can see that both MLs show higher TPR than GstLAL pipeline over given FPR ranges and it results in that the areas under curves of MLs are larger than the AUCs of $\ln L$. The performances of RF and NN are more or less similar to each other.

random guess.[5] We can understand this result as the ratio of signal samples having larger $\ln L$ is bigger than that of noise samples. In other words, we can say $\ln L^{(s)}_{\max}$ is bigger than $\ln L^{(n)}_{\max}$. From this result we know that $\ln L$ works as a proper ranking method in discriminating signal samples from noise samples as desired.

Also, one can see that two MLs show higher TPRs than the GstLAL pipeline over given FPR ranges and it results in the AUCs of MLs being about 4%–5% larger than the AUCs of GstLAL pipeline. When we focus on the TPR at the lower FPR region, specifically at the lowest FPR where $\ln L_{\rm th} = \ln L^{(n)}_{\max}$, MLs show 0.103–0.125 higher TPR than the TPR of GstLAL. This result shows that we could obtain highly probable signal samples 10.3%–12.5% more with MLs than GstLAL pipeline. In the consideration of a practical application of MLs to the low-latency search for GWs from binary mergers, the performance at lower FPR is important since FPR can be interpreted as the same analogy to the false alarm probability, which will be discussed in Sec. IV, and, eventually, identifying a GW signal candidate with sufficiently low false alarm probability will be connected to the declaration of the detection of a GW signal.

In the comparison between RF and NN, one can notice that the performance of each ML is similar to each other, despite RF showing about 0.73%–0.76% larger AUC and about 1.1%–1.7% higher TPR at the minimum FPR than NN depending on data. From this result, one can recognize that the classification performance of RF on the overall test data is the best.

B. Scatter plot

We need to examine how individual samples contribute to the resulted ROC curve. Thus, we investigate the contribution by looking at the correlation between the value of $\ln L$s of considered ranking methods and the feature parameters. We present example scatter plots in Figs. 2 and 3 drawn with selected feature parameters from the test data. In the scatter plot, the color bar on the right side shows normalized and unnormalized values of $\ln L$. Note that we use the chirp mass defined as

$$\mathcal{M} = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}} \tag{5}$$

instead of individual component masses, m_1 and m_2, for the horizontal axis because the chirp mass is one of the important parameters in describing characteristics of the evolution of the GW waveform generated from compact binary systems [51]. Also, since the SNR is one of the fundamental statistical quantities in judging the significance of a GW signal buried in noisy GW data, we especially select these two parameters for this example.

From Figs. 2 and 3, we see that signal samples of higher SNR and of higher chirp mass obtained higher $\ln L$s as expected from the ROC curve. In particular, for the distribution with respect to the chirp mass, we also see that signal samples are in the range of $[4.5, 45]\ M_\odot$, which is believed as the detectable chirp mass range for the BBH system by LIGO/Virgo. However, from the comparison of signal samples between GstLAL pipeline and MLs, GstLAL returns relatively lower $\ln L$s on samples having SNR $\lesssim 10$ which is a criterion for the SNR of the GW candidate signal. On the contrary, MLs return higher $\ln L$s even for those signal samples. This result means MLs can distinguish even less significant signal samples correctly

[5]The random guess is the case when a discriminator cannot distinguish a sample neither signal nor noise, i.e., the discriminator returns 0.5 for the probability of all signal and noise samples.

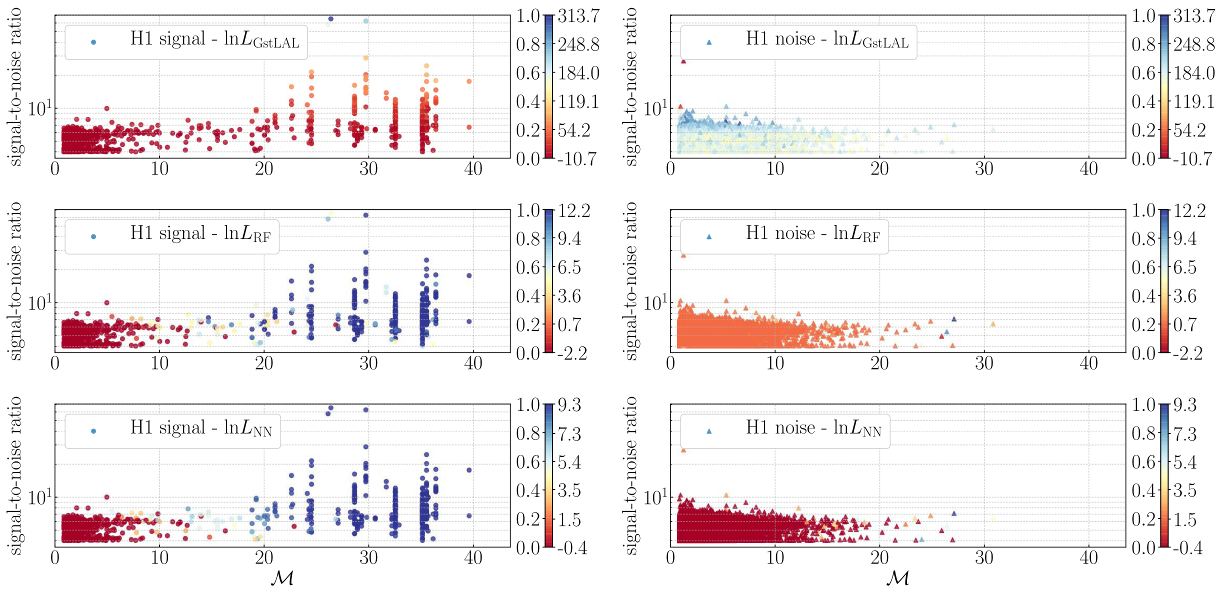


FIG. 2. Scatter plots of signal and noise samples of H1 data with respect to two selected feature parameters: SNR and the chirp mass, $\mathcal{M}$. The unit of $\mathcal{M}$ is given in $M_{\odot}$. The left and right columns show scatter plots of signal samples and noise samples, respectively. The color bar indicates the value of the normalized (left side) and unnormalized (right side) of $\ln L$s. One can see that MLs computed relatively higher $\ln L$ even for signal samples of lower SNR.

which may be disregarded as a candidate with GstLAL pipeline.

On the other hand, for noise samples, all ranking methods return relatively lower $\ln L$s than signal samples. However, in particular for H1 data, NN shows the best distinguishability than the other two methods in terms of normalized $\ln L$. Meanwhile, for L1 data, RF and NN shows similar distinguishability. Therefore we conclude that the distinguishability on individual samples is less effective in the computation of TPR and FPR of the ROC curve.

IV. SEARCH SENSITIVITY

In this section, we discuss the search sensitivity through the relation between the sensitivity range and

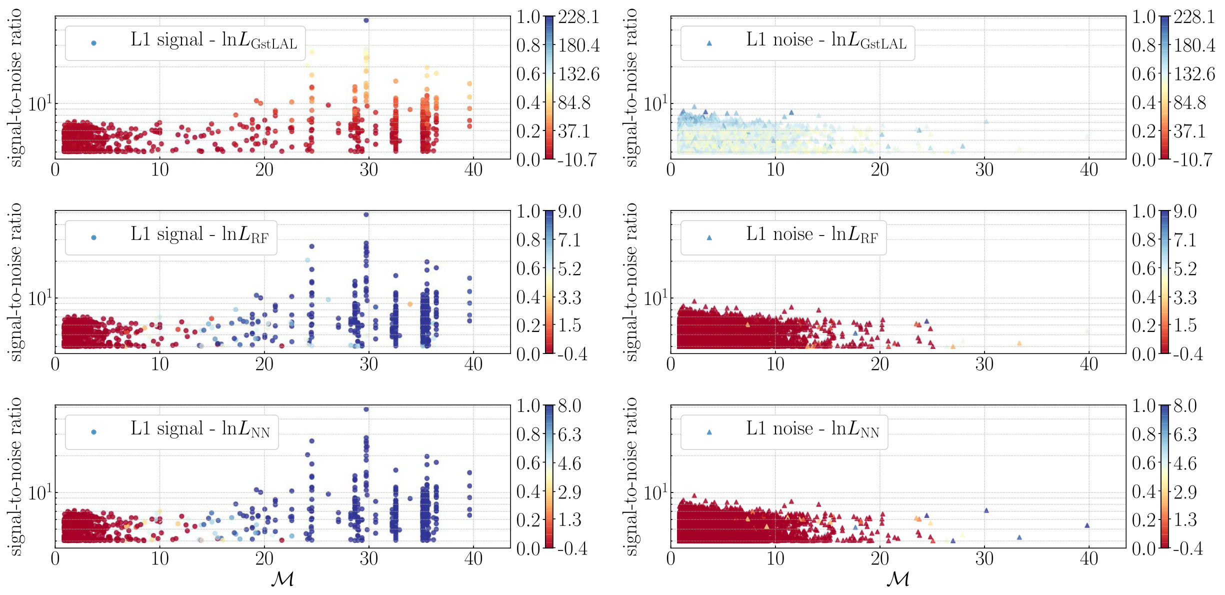


FIG. 3. Scatter plots of signal and noise samples of L1 data with respect to two selected feature parameters: SNR and the chirp mass, $\mathcal{M}$, as drawn in Fig. 2. One can see that the L1 data case shows similar to H1 data case but better discriminability on the RF result than the H1-RF result in Fig. 2.

the *false alarm rate* (FAR) [34] in order to suggest a practical application of MLs for searching GWs from compact binary systems. For this calculation, we refer to Sec. IV C of Ref. [11].

In general, FAR is defined as

$$\mathrm{FAR} \equiv \frac{1}{T}\int_{\ln L_{\mathrm{th}}}^{\infty} P(\ln L|n)d\ln L, \tag{6}$$

where T is the length of the data segment. We can use FAR for the determination of a threshold value for a ranking method and, eventually, can use the value to judge a detection of a GW signal. If we assume that the chance of background noises gaining higher $\ln L$ is very rare, we can write the FAR for the given $\ln L$ of a noise in an approximated form:

$$\mathrm{FAR} \approx \frac{P(\ln L \geq \ln L_{\mathrm{th}}|n)}{T}. \tag{7}$$

This approximation is valid for the consideration of this work too since it is shown that most of the noise samples obtain lower $\ln L$ than signal samples.

The numerator of Eq. (7) means the probability that noise data, n, get a high value of a ranking method. Thus it is called as *false alarm probability* (FAP) [34]. In this work, as discussed in Sec. III A, FPR can be interpreted in the same analogy of FAP when we compare Eq. (4) and the numerator of Eq. (7). However, at this moment, we change the $\ln L_{\mathrm{th}}$ in the expression of FAP to be $\ln L^{(s)}_{\mathrm{min}} \leq \ln L_{\mathrm{th}} \leq \ln L^{(s)}_{\mathrm{max}}$ instead of using $\ln L^{(n)}_{\mathrm{min}} \leq \ln L_{\mathrm{th}} \leq \ln L^{(n)}_{\mathrm{max}}$ since our interest is estimating the FAR of signal samples. Additionally, we follow the procedure for calculating Eq. (31) of Ref. [11] with the value of FPR in order to take account of the corrections which have been used in the conventional GstLAL pipeline for the computation of FAP.

For the calculation of the sensitivity range, we use the distance parameter which was used for the generation of the signal samples. We also adopt the definition of efficiency, ϵ, with the *found* sample given in Ref. [11]: when the estimated FARs of some signal samples are lower than a given fiducial FAR we call the signal sample *found* samples and the ratio of found samples to total number of signal samples *efficiency*. Then we compute the search volume such that

$$V = 4\pi \int_0^{\infty} \epsilon(r) l^2 dl, \tag{8}$$

where l is the distance to the source of GW and $\epsilon(l)$ is the efficiency at the distance l. Now the sensitivity range, R can be calculated by

$$R = \left(\frac{3V}{4\pi}\right)^{1/3}. \tag{9}$$

For the computation of the search volume, Eq. (8), we integrate the integrand with the *trapezoidal* method by varying the fiducial FAR. We refer to the sensitivity range, R, as the search sensitivity in this work and plot it with respect to the combined FAR in Fig. 4. The combined FAR is obtained by collecting an individual FAR computed with the data of each detector. In order to take account of the uncertainty in binary discrimination into signal or noise, we compute the lower and upper bounds, (ω^-, ω^+), of the Wilson confidence interval [52] with continuity correction [53]:

$$\omega^- = \max\left\{0, \frac{2y\hat{p} + z^2 - [z\sqrt{z^2 - 1/y + 4y\hat{p}(1-\hat{p}) + (4\hat{p}-2)} + 1]}{2(y+z^2)}\right\}, \tag{10}$$

$$\omega^+ = \min\left\{1, \frac{2y\hat{p} + z^2 + [z\sqrt{z^2 - 1/y + 4y\hat{p}(1-\hat{p}) + (4\hat{p}-2)} + 1]}{2(y+z^2)}\right\}, \tag{11}$$

where $\hat{p} = p/y$ is the fraction of found signal samples to the number, y, of total samples and z is the probit function. For Eqs. (10) and (11), if $\hat{p} = 0$, ω^- is taken as 0. On the other hand, if $\hat{p} = 1$, ω^+ is taken as 1.

From Fig. 4, we can see that we can detect more farther events with MLs for the given range of FAR than with GstLAL pipeline, in particular, at the lowest FAR, the central value of detectable range is ∼1.7 Gpc for both RF and NN while ∼1.4 Gpc for GstLAL pipeline. We see that the range of RF is slightly farther than that of NN. But, they are placed within 3σ uncertainty bounds of the Wilson confidence interval of each other. Therefore, we conclude their sensitivities are comparable and this result is consistent with the ROC curves discussed in Sec. III A.

V. SUMMARY AND DISCUSSION

Machine learning (ML) is known by its fast and accurate performance on identifying/classifying nonlinear multidimensional data of various fields. From several studies related to GW data analysis, applications of MLs have shown improved and/or comparable classification performances compared to conventional statistical approaches. Thus, in this work, we study the feasibility of whether we

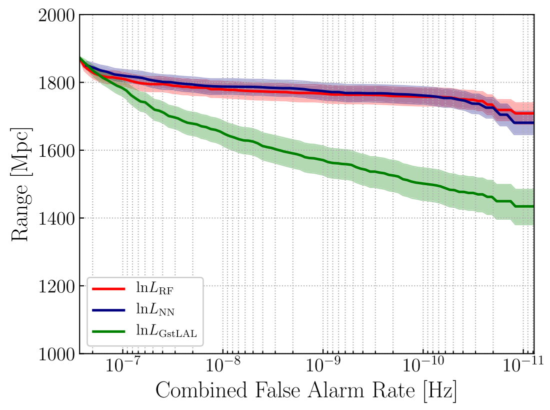


FIG. 4. Comparisons of the sensitivity in terms of detectable range in distance versus the combined false alarm rate (FAR). The red line indicates the result of RF, the navy line indicates the result of NN, and the lime line indicates the result of GstLAL pipeline. The shaded regions around each line show 3σ of the binomial confidence interval computed based on the Wilson method [52] with continuity correction [53]. One can see that the detectable range of MLs is relatively farther than that of GstLAL pipeline at the lower FAR region and it means we can identify an event occurred at farther distance with MLs.

can use the output of ML for scoring rank on candidate events for low-latency GW searches.

For this study, we consider two supervised MLs, random forest (RF) and neural network (NN). The mock data for GW150914 obtained by running GstLAL pipeline in offline mode is used as the signal sample. From the output of GstLAL pipeline, we extract six physical and statistical feature parameters for the configuration of input data for ML. With given data, we train considered MLs and test the classification performance of the rank of MLs to compare it to the conventional ranking method, the log-likelihood ratio, $\ln L$ of GstLAL pipeline. However, MLs return only probabilistic rank values in between 0 and 1 while $\ln L$ of GstLAL pipeline is unnormalized. Thus, to make a fair comparison, we first estimate the probability density function (PDF) for ranks of signal and noise samples separately and then compute $\ln L$ with the PDF.

It is known that ML should be trained with sufficiently large and nonbiased data for a successful application [54]. In general, training a ML with a large amount of data needs a long computational time from about hours to days to determine the most optimal combination for the hyperparameters of a ML. The training time depends on the size of train data such as the number of samples and the number of feature parameters. In this work, we find that training a ML with a set of train data of about $165{,}000(\text{number of samples})\times 6(\text{number of features})$ takes a few hours. Meanwhile, evaluating a test data or a new data requires a much shorter time than time for training: from our study, the evaluation with a set of test data of roughly $45,000(\text{number of samples})\times 6(\text{number of features})$ dimensions can be done in the order of tens of milliseconds. Thus, we can see the positive prospect of applying ML for the low-latency GW search in terms of the analysis speed.

We investigate the classification performance of MLs through a couple of figures of merit. In this work, we choose the receiver operating characteristic (ROC) curve to see overall performance for all tested samples and the scatter plot to see the contribution to an individual sample. From the ROC curve and the area under the curve, we can observe that MLs show better performance on classifying more signal samples from noise samples than GstLAL pipeline. The result on individual samples is also studied through the scatter plot to try to see the correlation between the output value of a given ranking method and selected feature parameters, e.g., signal-to-noise ratio (SNR) and chirp mass. We find that signal samples of relatively higher $\ln L$s have higher SNR and chirp masses within expected chirp mass range for binary black hole mergers. Thus we are convinced that the resulting higher $\ln L$ values of given ranking methods are correlated to the tested feature parameters.

For the difference in performance between RF and NN, one may suspect that the hyperparameters of NN might be less optimized than RF since we empirically selected the hyperparameters of NN without automated determination as discussed in Appendix B. However the most optimal choice on the hyperparameter depends on the input data for training: if we train MLs with different training data, the choice on the set of optimal hyperparameters will be also changed. On top of that, it is hard to think the data used in this work can represent the general property of all possible BBH systems. Thus, conducting more fine-tuning on the optimal hyperparameters for NN with the data used in this work is out of the scope of this kind of feasibility study and we admit the difference between RF and NN is placed in the acceptable range.

We compare the sensitivity in terms of the detectable range with respect to the approximated false alarm rate (FAR) too. In specific, since the false positive rate (FPR) of the ROC curve can be translated to the false alarm probability, the FPR of the ROC curve is also used in computing the approximated FAR. From the sensitivity plot, we can see that MLs are more sensitive than GstLAL pipeline, that is, it would be possible to detect farther events with MLs beyond the upper limit of the detectable range of GstLAL pipeline. Therefore we conclude that using output of ML can be an alternative ranking method to the conventional ranking method of GstLAL pipeline for obtaining improved detection significance and it is worth considering ML as a new ranking method for future low-latency searches for GWs from compact binary mergers.

In this work, we constrained the origin of feature parameters of input data for ML only to the information obtained from the GstLAL pipeline for simplicity. However, it is also possible to consider collecting transient noise information, which is used for the GW data quality measurement, along with the current methodology. In the future work, therefore, we will discuss the practical implementations such as training MLs with combining the current feature parameters and transient noise information into the input data. Next, we will build up the strategy and the framework for an on-line GW inspiral search pipeline which implements ML as its ranking method.

However we admit that this approach is rather weak in interpretability: it is not easy to clearly understand how the model results in the better result with less information than the conventional method. This point is one of the differences from the conventional approach because it is built on statistically reliable considerations and, eventually, has strong interpretability. The interpretability is another critical point for judging the confidence of a detection. Therefore, for the practical implementation, we may also need to design an additional method to make the classification model to be interpretable.

ACKNOWLEDGMENTS

The authors thank the LIGO Scientific Collaboration for the use of the mock data for GW150914 and corresponding background data. K. K. would like to specially thank to J. J. Oh, S. H. Oh, E. J. Son, Y.-M. Kim, W. Kim, J. Lee, S. Caudill, and K. Cannon for fruitful and constructive discussions. The work described in this paper was partially supported by grants from the Research Grants Council of the Hong Kong (Projects No. CUHK 14310816 and No. CUHK 24304317) and the Direct Grant for Research from the Research Committee of the Chinese University of Hong Kong. The work of K. K. was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2020R1C1C1005863).

APPENDIX A: HYPERPARAMETERS OF RF

For RF, we run a module, GridSearchCV embedded in Scikit-Learn for searching optimal hyperparameters. The core of this module is the *k-fold cross validation* method: conducting k-times validation tests on k different validation subsets, which are prepared by shuffling all train data and then dividing evenly into k subsets, with a given combination of hyperparameters. Each validation test is done by evaluating one of the k subsets as a test subset based on the trained ML which is trained with remaining subsets. This module computes the *averaged accuracy*:

$$\text{average accuracy} \equiv \frac{\text{number of samples}(y_{\text{true}} = y_{\text{pred}})}{\text{total number of samples in a subset}} \tag{A1}$$

as the final output of each test for RF. In Eq. (A1), y_{true} and y_{pred} denote, respectively, the original class and the predicted class of a sample. At last, a combination of hyperparameters which gives the highest averaged accuracy is selected as the most optimal set of hyperparameters. Used entries of selected hyperparameters for running this module are tabulated in Table III. One can find the description of each hyperparameter from Ref. [55].

In this work, we conduct the run of GridSearchCV with $k = 3$ and the determined optimal hyperparameters for given data are summarized in Table IV. From this table, one can see that some hyperparameters are the same for different data. It means those values are the most optimal value amongst tested entries for the type of data of this work. Therefore, if we do not change the selected six feature parameters for a similar type of data, we can fix the values of those hyperparameters and alter remains for the determination of optimal hyperparameters.

TABLE III. Tested entries for hyperparameters of RF in running GridSearchCV.

Hyperparameter	Entry
`n_estimators`	50, 100, 200
`criterion`	Gini, entropy
`max_features`	2, 4, 6
`min_samples_split`	2, 3, 4, 5
`max_depth`	None, 10, 30, 50

TABLE IV. Empirically determined optimal hyperparameters of RF by running GridSearchCV. One can see that some hyperparameters are the same for different data.

Hyperparameter	Data	Optimal
`n_estimators`	H1	50
	L1	50
`criterion`	H1	Entropy
	L1	Entropy
`max_features`	H1	4
	L1	4
`min_samples_split`	H1	3
	L1	4
`max_depth`	H1	30
	L1	10

APPENDIX B: HYPERPARAMETERS OF NN

Unlike RF, we set the hyperparameters of NN empirically because there is no available module for grid search in TensorFlow.[6] We set them to be the same for all considered data for convenience. In addition to the hyperparameters for

[6]It is also known that it is hard to consider such grid search model for DNN because there are too many hyperparameters to be tuned [54]. However, it is not impossible at all if we use an automated method such as DeepHyper [56]. Despite the availability of the automated hyperparameter search method, we do not implement it in this work because we could get satisfactory performance with the empirically determined hyperparameters.

TABLE V. Hyperparameters for NN.

Hyperparameter	Value
Layers (nodes)	1 input (6 nodes) 4 hidden (32 nodes for each) 1 output (1 node)
Learning rate	0.01%
Regularization	$L2$ with 0.01%
Dropout	10%
Activation function	ReLU
Cost function	Cross entropy with Softmax
Batch size	1024

the topology of a NN, in order to avoid overestimation (or overfitting equivalently), we adopt $L2$ *regularization* to constrain a NN's connection weights and *dropout* [57] to remove potential dependency on certain nodes of the given network. The *rectified linear units* (ReLU) function [58] is used for the activation function between nodes in two adjacent layers. For the cost function, which measures the error between the target value, 1 for signal and 0 for background, and the output value in between 0 and 1 of the output node, the *Cross entropy* function is implemented by taking the output probability computed from the *Softmax* function as the input probability of the Cross entropy function. Finally, we also consider *Batch normalization* [59] to properly minimize the cost function in the *back-propagation* [60] process. All of these hyperparameters are summarized in Table V as well.

APPENDIX C: PROBABILITY DENSITY ESTIMATION

In order to compute log-likelihood ratio, we estimate the probability density functions (PDFs) of signal and noise samples for the numerator and the denominator respectively of Eq. (2). The estimation of PDFs of resulted ranks of evaluation samples is done by using the *kernel density estimation* (KDE) method of Scikit-Learn with Gaussian kernel and an empirically determined optimal bandwidth of 0.03. For this estimation, we compute normalized density distribution for the ranks of evaluation samples first and apply KDE with given bandwidth. The resulted PDFs for signal and noise samples of each test data using ranks from each ML are shown in Fig. 5. From these plots and Eq. (2), we can expect that we can obtain smaller $\ln L$ with lower ranks and bigger $\ln L$ with higher ranks.

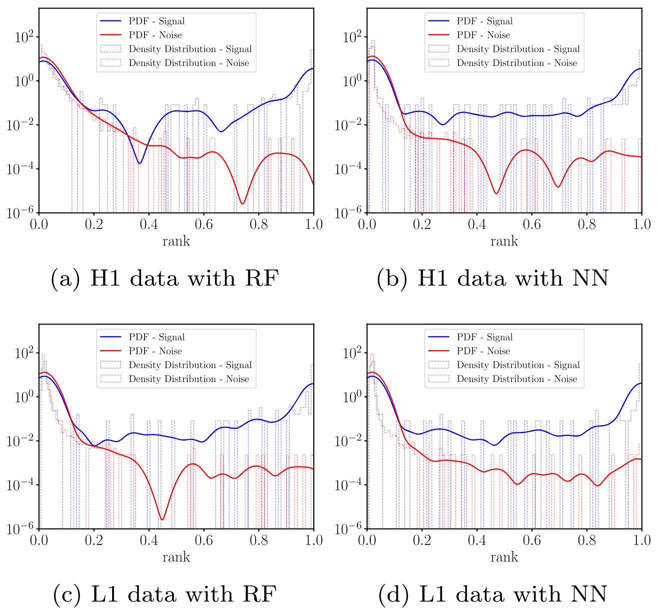


FIG. 5. Estimated probability density functions (PDFs) for each data with studied MLs. For the PDF estimation, we use the kernel density estimation method of Scikit-Learn with Gaussian kernel and bandwidth of 0.03. The blue- and red-solid lines are the estimated PDFs for signal samples and for noise samples, respectively. The blue- and red-dashed boxes show the normalized density distribution of signal and noise samples, respectively, used for the estimation of PDFs.

[1] 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).
[2] 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 (2015).
[3] B. P. Abbott *et al.*, Phys. Rev. Lett. **119**, 024025 (2017).
[4] A. von Kienlin, C. Meegan, and A. Goldstein, GCN Circ. **21520**, https://gcn.gsfc.nasa.gov/gcn3/21520.gcn3.
[5] C. A. Meegan *et al.*, Astrophys. J. **702**, 791 (2009).
[6] B. P. Abbott *et al.*, Astrophys. J. Lett. **848**, L12 (2017).
[7] R. Margutti *et al.*, arXiv:1812.04051.
[8] Ž. Ivezić *et al.*, Astrophys. J. **873**, 111 (2019).
[9] K. Somiya, Classical Quantum Gravity **29**, 124007 (2012).
[10] B. R. Iyer, Souradeep, T. C. S. Unnikrishnan, S. Dhurandhar, S. Raja, and A. Sengupta, Ligo-India, Proposal of the consortium for Indian initiative in gravitational-wave observations (IndIGO), IndIGO Consortium, LIGO-M1100296, 2011.
[11] C. Messick *et al.*, Phys. Rev. D **95**, 042001 (2017).
[12] J. Luan, S. Hooper, L. Wen, and Y. Chen, Phys. Rev. D **85**, 102002 (2012).
[13] J. Abadie *et al.*, Astron. Astrophys. **541**, A155 (2012).
[14] S. A. Usman *et al.*, Classical Quantum Gravity **33**, 215004 (2016).

[15] T. Adams, D. Buskulic, V. Germain, G. M. Guidi, F. Marion, M. Montani, B. Mours, F. Piergiovanni, and G. Wang, Classical Quantum Gravity **33**, 175012 (2016).
[16] S. Klimenko, I. Yakushin, A. Mercer, and G. Mitselmakher, Classical Quantum Gravity **25**, 114029 (2008).
[17] GraceDB, https://gracedb.ligo.org.
[18] Gamma-ray Coordinates Network, https://gcn.gsfc.nasa.gov.
[19] R. Biswas *et al.*, Phys. Rev. D **88**, 062003 (2013).
[20] T. S. Adams, D. Meacher, J. Clark, P. J. Sutton, G. Jones, and A. Minot, Phys. Rev. D **88**, 062006 (2013).
[21] S. Rampone, V. Pierro, L. Troiano, and I. M. Pinto, Int. J. Mod. Phys. C **24**, 1350084 (2013).
[22] K. Kim, I. W Harry, K. A Hodge, Y.-M. Kim, C.-H. Lee, H. K. Lee, J. J. Oh, S. H. Oh, and E. J. Son, Classical Quantum Gravity **32**, 245002 (2015).
[23] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. K. Katsaggelos, S. L. Larson, T. K. Lee, C. Lintott, T. B. Littenberg, A. Lundgren, C. Østerlund, J. R. Smith, L. Trouille, and V. Kalogera, Classical Quantum Gravity **34**, 064003 (2017).
[24] S. J. Kapadia, T. Dent, and T. Dal Canton, Phys. Rev. D **96**, 104015 (2017).
[25] N. Mukund, S. Abraham, S. Kandhasamy, S. Mitra, and N. S. Philip, Phys. Rev. D **95**, 104059 (2017).
[26] J. Powell, D. Trifir, E. Cuoco, I. S. Heng, and M. Cavagli, Classical Quantum Gravity **32**, 215012 (2015).
[27] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).
[28] D. George and E. A. Huerta, Phys. Rev. D **97**, 044039 (2018).
[29] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Phys. Rev. D **100**, 063015 (2019).
[30] GstLAL, https://www.lsc-group.phys.uwm.edu/daswg/projects/gstlal.html.
[31] GStreamer, https://gstreamer.freedesktop.org.
[32] LALSuite, https://www.lsc-group.phys.uwm.edu/daswg/projects/lalsuite.html.
[33] B. J. Owen and B. S. Sathyaprakash, Phys. Rev. D **60**, 022002 (1999).
[34] K. Cannon, C. Hanna, and D. Keppel, Phys. Rev. D **88**, 024025 (2013).
[35] S. Sachdev *et al.*, arXiv:1901.08580.
[36] L. Breiman, Mach. Learn. **45**, 5 (2001), and references therein.
[37] W. S. McCulloch and W. Pitts, Bull. Math. Biophys. **5**, 115 (1943).
[38] Scikit-Learn, http://scikit-learn.org.
[39] TensorFlow, https://www.tensorflow.org.
[40] R. Perna, D. Lazzati, and B. Giacomazzo, Astrophys. J. **821**, L18 (2016).
[41] A. Loeb, Astrophys. J. **819**, L21 (2016).
[42] N. C. Stone, B. D. Metzger, and Z. Haiman, Mon. Not. R. Astron. Soc. **464**, 946 (2017).
[43] S. E. D. Mink and A. King, Astrophys. J. **839**, L7 (2017).
[44] B. Mckernan *et al.*, Astrophys. J. **866**, 66 (2018).
[45] V. Connaughton *et al.*, Astrophys. J. **826**, L6 (2016).
[46] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. X **6**, 041015 (2016); **8**, 039903(E) (2018).
[47] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. D **93**, 122003 (2016).
[48] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Astrophys. J. **833**, L1 (2016).
[49] A. Taracchini *et al.*, Phys. Rev. D **89**, 061502 (2014).
[50] M. Was, M.-A. Bizouard, V. Brisson, F. Cavalier, M. Davier, P. Hello, N. Leroy, F. Robinet, and M. Vavoulidis, Classical Quantum Gravity **27**, 015005 (2010).
[51] J. D. E. Creighton and W. G. Anderson, *Gravitational-Wave Physics and Astronomy* (Wiley-VCH, New York, 2011).
[52] E. B. Wilson, J. Am. Stat. Assoc. **22**, 209 (1927).
[53] R. G. Newcombe, Stat. Med. **17**, 857 (1998).
[54] A. Géron, *Hands-On Machine Learning with Scikit-Learn & TensorFlow* (O'Reilly, Sebastopol, CA, 2017).
[55] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, for hyperparameters of Random Forest.
[56] P. Balaprakash, M. Salim, T. D. Uram, V. Vishwanath, and S. M. Wild, DeepHyper: Asynchronous hyperparameter search for deep neural networks, in *IEEE 25th International Conference on High Performance Computing (HiPC), Bengaluru, India, 2018*(IEEE Computer Society, Los Alamitos, CA, 2018), pp. 42–51.
[57] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, arXiv:1207.0580.
[58] V. Nair and G. Hinton, in *ICML'10 Proceedings of the 27th International Conference on International Conference on Machine Learning* (Omnipress, Madison, WI, 2010), pp. 807–814.
[59] S. Ioffe and C. Szegedy, arXiv:1502.03167.
[60] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning internal representation by error propagation, in *Parallel Distributed Processing. Exploration of the Microstructure of Cognition*, edited by D. E. Rumelhard and J. L. McCleland (MIT Press, Cambridge, MA, 1986), pp. 318–362.