Detection of gravitational waves using Bayesian neural networks

Yu-Chiung Lin and Jiun-Huei Proty Wu[*]

Department of Physics, Institute of Astrophysics, and Center for Theoretical Physics, National Taiwan University, Taipei 10617, Taiwan

(Received 10 July 2020; accepted 2 March 2021; published 23 March 2021)

We propose a new model of Bayesian neural networks to not only detect the events of compact binary coalescence in the observational data of gravitational waves (GW) but also identify the full length of the event duration including the inspiral stage. This is achieved by incorporating the Bayesian approach into the convolutional, long short-term memory, fully connected deep neural network classifier, which integrates together the convolutional neural network (CNN) and the long short-term memory recurrent neural network (LSTM). Our model successfully detect all seven binary black hole events in the LIGO Livingston O2 data, with the periods of their GW waveforms correctly labeled. The ability of a Bayesian approach for uncertainty estimation enables a newly defined 'awareness' state for recognizing the possible presence of signals of unknown types, which is otherwise rejected in a non-Bayesian model. Such data chunks labeled with the awareness state can then be further investigated rather than overlooked. Performance tests with 40,960 training samples against 512 chunks of 8-second real noise mixed with mock signals of various optimal signal-to-noise ratio $0 \leq \rho_{\rm opt} \leq 18$ show that our model recognizes 90% of the events when $\rho_{\rm opt} > 7$ (100% when $\rho_{\rm opt} > 8.5$) and successfully labels more than 95% of the waveform periods when $\rho_{\rm opt} > 8$. The latency between the arrival of peak signal and generating an alert with the associated waveform period labeled is only about 20 seconds for an unoptimized code on a moderate GPU-equipped personal computer. This makes our model possible for nearly real-time detection and for forecasting the coalescence events when assisted with deeper training on a larger dataset using the state-of-art HPCs.

DOI: 10.1103/PhysRevD.103.063034

I. INTRODUCTION

Since the first detection of gravitational waves (GWs) on September 14th, 2015 [1], the Advanced Laser Interferometer Gravitational Wave Observatory (aLIGO) [2], later joined by the Advanced Virgo [3] in 2017, has detected thirteen coalescence events for binary black holes (BBHs) [1,4–12] and two for binary neutron stars [13,14] during its O1, O2 and ongoing O3 observation runs. The triumph also comes with a remarkable milestone where the electromagnetic (EM) counterparts of the binary neutron star coalescence event GW170817 [13] was discovered [15–17]. The success of LIGO and Virgo has opened a new era of multi-messenger astronomy, allowing for independent measurement of Hubble constant [18] and constraints on theoretical models such as cosmic strings [19]. In addition, KAGRA [20], a GW observatory in Japan, started observation in February 2020 and another GW detector located in India [21] is also about to join the network in order to increase both the overall sensitivity and the precision in sky source locations. While more detectors join the network, the need for techniques of real-time detection has become more pressing not only for accurate determination of sky source locations but also for the counterpart observations such as those for EM signals [22]. For example, the searches for EM counterparts such as the kilonovae and short-gamma-ray bursts [23] can improve the accuracy of source parameter estimation while increasing the confidence for GW detections [24,25], and help break the modeling degeneracy of binary properties [24,26] so as to understand better the nature of binary systems and their host galaxies [27]. However, the commonly used match-filtering techniques [28–33] are computationally expensive, making it a great challenge for real-time detection.

Recently the deep learning technique based on artificial neural networks [34] is considered as a promising alternative to the matched-filtering method. Various types of deep neural networks (DNNs) such as the convolutional neural network (CNN) [35] and the long short-term memory recurrent neural network (LSTM) [36–38] have shown great potential in the framework of GW research, especially for real-time detection [39–45], parameter estimation [39,40,43,46], glitch recognition [47–50] and data denoising [51–54]. However, despite their potential prospect, the DNN models usually suffer from overfitting and thus are difficult to be generalized for making reliable

[*]jhpw@phys.ntu.edu.tw

predictions in face of the data that are out of distribution. In addition, most DNN models are deterministic in a way that they offer only one rigid prediction for each given set of input. A deterministic model can hardly provide information about the uncertainty in its predictions. Therefore the problems of overfitting and lack of uncertainty estimation make the DNN models unreliable, sometimes giving overconfident predictions on out-of-distribution data [55].

In order to deliver the uncertainty information, we need to convert the traditional deterministic model into a probabilistic model. In the field of deep learning, the Bayesian neural network (BNN) [56,57], which updates its weights via Bayes' rule, is a potential choice for this purpose. The BNN technique is not only capable of providing uncertainty estimation but also resistant to overfitting. In addition, it can be trained with a rather small dataset. The uncertainty estimation is particularly useful in face of the out-of-distribution data and could assist further training for data augmentation. Although the BNN is more computationally expensive than the deterministic DNN, it is totally feasible when assisted with the recent-year advances in both the hardware and the approximation algorithms [58–63].

In this paper we use the variational inference (VI) approximation [59–61,64] to construct a convolutional, long short-term memory, fully connected deep neural network (CLDNN) model [65]. The incorporation of the sliding-window search scheme enables us to identify the time period of the GW waveform in a coalescence event. This feature distinguishes our model from other CNN models for GW detection in literature. The combination of a Bayesian approach and the CLDNN is also a unique feature of our model. Based on the model prediction and its estimated uncertainty, our Bayesian model can rapidly flag each of the sequential time windows with a trigger state, a noise state, or a state that needs further attention. The estimated uncertainty can also serve as a reference for the significance level of a prediction. Our model is not to replace the matched-filtering search or other models of parameter estimation but to serve as a prior process for efficiently identifying the time windows of signals and those which may contain new-type signals. Our model also enables the possibility for nearly real-time detection, which is unlikely to be feasible for the usually time-consuming matched-filtering search.

We organize this paper as follows. In Sec. II, we introduce our BNN-based method, including the model architecture, data preparation, training procedure, and the flagging strategy. In Sec. III, we demonstrate with uncertainty estimation the capability of our model in detecting gravitational waves, first with benchmarking tests and then against the real data from LIGO. In Sec. IV we discuss several critical issues in our model and its potential for event prediction. Finally we conclude our work in Sec. V.

II. BAYESIAN NEURAL NETWORK FOR GRAVITATIONAL WAVE DETECTION

A. Bayesian neural network

The Bayesian neural network [56,57] is the deep neural network in which its hidden units use probability distributions, instead of point values, as weights and biases. The BNN can provide uncertainty estimation about its prediction and handle overfitting. The parameters in a BNN model are initialized with prior distributions $p(W)$, where W represents the model weight. When trained with a given dataset $\mathcal{D}$, these prior distributions are updated to posterior distributions $p(W|\mathcal{D})$ via the Bayes' rule:

$$p(W|\mathcal{D}) = \frac{p(\mathcal{D}|W)p(W)}{p(\mathcal{D})}, \tag{1}$$

where $p(\mathcal{D}|W)$ is the likelihood and $p(\mathcal{D})$ is the model evidence. One can then obtain the predictive distribution, $p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$, for a new input $\mathbf{x}^*$ as

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^*|\mathbf{x}^*, W)p(W|\mathcal{D})dW. \tag{2}$$

However, for most of the modern neural networks the posterior $p(W|\mathcal{D})$ cannot be analytically calculated or efficiently sampled due to the enormous number of parameters in the model. To tackle this problem we employ the variational inference (VI) [59,66], which approximates the true posterior $p(W|\mathcal{D})$ with some tractable distributions $q_{\boldsymbol{\theta}}(W)$ that can be fully parametrized by $\boldsymbol{\theta}$. For example, the commonly used Gaussian distribution can be parametrized by the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$. The training goal is to minimize the negative evidence lower bound (ELBO): [59,60]

$$\mathcal{F}(\boldsymbol{\theta}) = -\mathop{\mathbb{E}}_{W\sim q_{\boldsymbol{\theta}}}[\log p(\mathcal{D}|W)] + \mathrm{D}_{\mathrm{KL}}(q_{\boldsymbol{\theta}}||p), \tag{3}$$

where the first term is the expectation value of negative log likelihood and the second term is the Kullback-Leibler divergence [67,68] between distributions $q_{\boldsymbol{\theta}}$ and p. In classification tasks, the first term is equivalent to the cross-entropy loss.

Within the framework of currently available machine learning tools, the VI estimator used in BNNs is usually achieved by perturbing the weights and biases [59–62]. At each forward pass, the model weights and biases are randomly sampled from the distributions $q_{\boldsymbol{\theta}}(W)$. If the distributions are independent Gaussian distributions, the gradients can be computed using back propagation [62]. Thus the predictive distributions can be approximated by propagating the input $\mathbf{x}^*$ through the model multiple times and then taking the average:

$$p(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D}) \approx \int_W p(\mathbf{y}^*|\mathbf{x}^*,W)q_{\boldsymbol{\theta}}(W)dW$$
$$\approx \frac{1}{N}\sum_{i=1}^{N} p(\mathbf{y}^*|\mathbf{x}^*,W_i),\ W_i \underset{i.i.d.}{\sim} q_{\boldsymbol{\theta}}(W), \quad (4)$$

where N is the number of Monte-Carlo samples. However, for a minibatch training the samples in a batch usually share the same weight perturbation for computational efficiency, and this will induce correlation between the gradients and thus make the model hard to converge due to the high variance in the gradient estimation [61]. In addition, it is hard to conduct inference for unknown samples of shared weight perturbations because we are forced to propagate one sample through the model at a time in renewing the weight perturbation. To cope with this, Ref. [61] proposed a flipout estimator that applies a random sign matrix on the weight perturbation matrix for each sample, so as to achieve a pseudo-independent weight sampling for each sample in a mini batch. The flipout estimator does not limit the variance reduction effect during training when trained with a large batch, and it also enables the Monte-Carlo (MC) sampling with minibatch prediction and thus speeds up the prediction process. In this work we use the Bayes by backprop (BBB) VI method proposed in Ref. [60] in combination with this flipout estimator [61] to train our model.

Uncertainty estimation is an important feature of the BNN, making it more robust for unknown input than the deterministic neural networks. For a softmax classifier, the predictive uncertainty can be defined as the covariance of the predictive distribution [69,70]:

$$\mathbf{U} = \frac{1}{N}\sum_{i=1}^{N}(\mathrm{diag}(\mathbf{p}_i) - \mathbf{p}_i^{\otimes 2}) + \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i - \bar{\mathbf{p}})^{\otimes 2}, \quad (5)$$

where $\mathbf{p}_i$ is the predictive vector of the ith MC sample, $\bar{\mathbf{p}}$ is the mean predictive vector, and for a given vector $\mathbf{v}$ we define the notations $\mathbf{v}^{\otimes 2} = \mathbf{v}\mathbf{v}^{\mathrm{T}}$ and $\mathrm{diag}(\mathbf{v})$ being a diagonal matrix with elements from $\mathbf{v}$. The first term in Eq. (5) is called the aleatoric uncertainty, which captures inherent randomness of the prediction $\mathbf{p}_i$. The second term is called the epistemic uncertainty, which originates from the variability of W given the dataset $\mathcal{D}$. In binary classification the off-diagonal elements in $\mathbf{U}$ normally provide no useful information so we focus only on the diagonal elements, which are the variances of the predictions in each class. Therefore the uncertainty of a prediction on the kth class can be simplified as [71]:

$$u_k = \frac{1}{N}\sum_{i=1}^{N}(p_{k,i} - p_{k,i}^2) + \frac{1}{N}\sum_{i=1}^{N}(p_{k,i} - \bar{p}_k)^2. \quad (6)$$

Because empirically we have $u_k < 0.25$, we intuitively define a confidence score for the kth class as [71]:

$$c_k = 1 - 2\sqrt{u_k}, \quad (7)$$

which indicates how confident the model is in its prediction.

B. Architecture of our model

It would be extremely useful if a neural network model for GW detection could also provide information about the duration of a coalescence event, which normally lies between less than a second and tens of seconds depending on the component masses. With the duration information, we would be able to dramatically reduce the search space for component masses and thus speed up the match filtering process. Recent work in Ref. [72] and Ref. [41] proposed fully convolutional neural networks based on the structure of WaveNet [73] that accepts input data of various lengths but their model only triggers the alarm around the GW signal peak while providing no information about the signal duration. Here we propose a model that could provide the duration information.

Our approach is a CLDNN model [65], which takes advantage of the complementarity of CNNs, LSTMs and DNNs by combining them into one unified architecture (see Sec. I for full names). The LSTM [36,38] is a type of recurrent neural networks (RNNs) [74]. A RNN layer consists of several cells, where the hidden units in the current time step have additional recurrent connections to those in next time step [75]. The output of these recurrent connections is called the "hidden state," which enables the cell to keep the memories from the previous time step. The main function of the LSTM layer is to use an additional "cell state" to further track the long-term memories. This characteristic of the LSTM enables us to detect not only the peak signal of a coalescence event but also the earlier signals from the inspiral stage. We also incorporate the advantage addressed in Ref. [41] that when one chooses the half-length of the CNN time window to be the stride size in a sliding-window approach, one could always capture the main part of the waveform in the following time windows even if it lies only partially within the current time window, and in turn the LSTM structure can correlate these windows to make meaningful predictions.

As a first step to deliver the above, we choose the input format of our model to be the strain data sliced into time windows (time steps) of fixed length, each of which is 50% overlapped with its neighbors. The details of data generation will be described in Sec. II C. There is essentially no limit on the number of windows so our model is suitable for data of any length. Nevertheless in each operation the size of windows needs to be fixed due to the structural requirement of the LSTM and the fully connected layers.

Our model is composed of three main sequential sectors: the CNN, LSTM, and fully connected (FC). Fig. 1 shows the structure. The CNN sector has four convolutional blocks, each with four layers: Bayesian convolution,

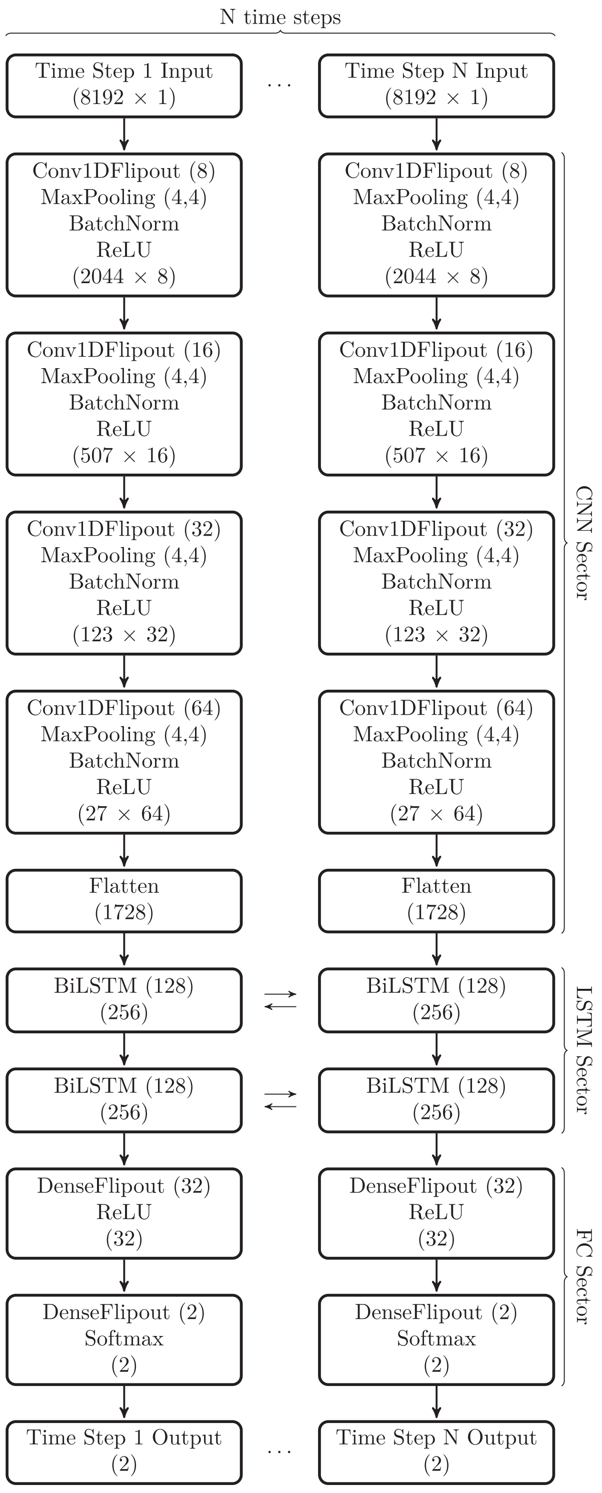
N time steps
Time Step 1 Input (8192 × 1)
Time Step N Input (8192 × 1)
Conv1DFlipout (8) MaxPooling (4,4) BatchNorm ReLU (2044 × 8)
Conv1DFlipout (16) MaxPooling (4,4) BatchNorm ReLU (507 × 16)
Conv1DFlipout (32) MaxPooling (4,4) BatchNorm ReLU (123 × 32)
Conv1DFlipout (64) MaxPooling (4,4) BatchNorm ReLU (27 × 64)
Flatten (1728)
BiLSTM (128) (256)
BiLSTM (128) (256)
DenseFlipout (32) ReLU (32)
DenseFlipout (2) Softmax (2)
Time Step 1 Output (2)
Time Step N Output (2)
CNN Sector
LSTM Sector
FC Sector

FIG. 1. The structure of our Bayesian CLDNN model. The dimensions of the outputs from each block are indicated at the bottom of each block.

max pooling [76], batch normalization [77], and rectified linear unit (ReLU) activation [78]. The Bayesian convolution layer uses several filters to extract different features from the input, and each filter has a small set of shared weight distributions called "kernel" to perform the convolution (or cross correlation) [35]. Different dilation rates are used to determine the separation of kernel weights in the space. The four Bayesian convolution layers here have 8, 16, 32, and 64 filters, each with the kernel size of 16, 8, 8, and 8 and the dilation rate of 1, 2, 2, and 2, respectively. The stride size of the kernels in all Bayesian convolution layers is fixed to 1. The max pooling layer helps us downsample the data by picking up the maximum value in the pooling window. All the pooling layers have a window size of 4, sliding with a stride size of 4. After the convolution the data are flattened and then passed into the LSTM sector.

The LSTM sector has two bidirectional Bayesian LSTM layers, each with 128 hidden units (weights) in each direction. Finally, the FC sector contains two Bayesian FC layers, with 32 and 2 hidden units respectively. The first Bayesian FC layer is followed by an ReLU activation layer, and the second by a softmax activation layer in order to confine the output values between 0 and 1. All Bayesian layers employ both the VI [59,60,64] and the flipout technique [61] to generate pseudo-independent weight and bias perturbations in the hidden layers.

Our model has a total of around 4.65 million parameters. Its outputs are the class results (class 0 for noise and class 1 for signal), each attached with a confidence score between 0 and 1 as a reference to judge on the existence of the GW signals from coalescence events in the corresponding time windows.

C. Data preparation

1. Real noise data

In this work we use the LIGO Livingston O2 data segments provided by the Gravitational Wave Open Science Center [79] (GWOSC) as our background noise of GW injection. We randomly select 15 data segment files from the first few days of the observation for training,[1] 5 files from the final day for validation,[2] and 8 files from the middle for testing the model.[3] All data segment files are 4096 seconds in length and have a quality of at least `CBC_CAT3` = 100. They do not contain any of the events or marginal triggers published by LIGO [9]. We downsample the sample rate of the strain data from 16384 to 8192 Hz in order to save memory.

2. Simulated templates of GW signals

We use the software packages PyCBC [29,30] and LALSuite [80] to generate GW signals of quasicircular, nonspinning BBH coalescence with the effective-one-body model SEOBNRv4T [81]. The signals are simulated with a sample rate of 8192 Hz and a cut-off frequency of 20 Hz.

[1]December 1st to 3rd, 2016.
[2]August 25th, 2017.
[3]April 9th and 10th, 2017.

The choice of component masses is similar to those in Ref. [39,40]. For the training dataset the BBH component masses range between 5 $M_{\odot}$ and 75 $M_{\odot}$ in steps of 1 $M_{\odot}$, with a mass ratio of $q = M_1/M_2 \leq 10$. For the validation dataset the masses are offset by 0.5 $M_{\odot}$ with respect to those in the training dataset. This offset helps ensure that the network is not overfitting by memorizing only the inputs shown to it without learning to generalize to new inputs [39,40]. For the testing dataset the masses are the collection of those values used in the above two datasets. In order to make our simulated samples more realistic, we also incorporate the GPS time stamps (of the associated real noise; see below) and a set of randomly generated right ascensions and declinations (for the signal part) into the samples following the convention of LIGO L1 detector. This in turn means that every simulated waveform in our dataset is different.

3. Data generation

In generating various datasets for training, validation, and testing, we use the real noise randomly picked up from the LIGO dataset (Sec. II C 1) for every sample, and inject the simulated signals (Sec. II C 2) into half of the samples for each dataset. To manipulate the signal-to-noise ratio (SNR) for the samples that contain signals, we precalculate the power spectral density (PSD) of the noise $S_n(f)$ and tune the strength of the GW signal h so that the optimal SNR, ρ_{opt}, defined by [82]

$$\rho_{\text{opt}}^2 = 4 \int_0^{\infty} df \frac{|\tilde{h}(f)|^2}{S_n(f)}, \tag{8}$$

lies within the desired range (see below). Here $\tilde{h}(f)$ is the Fourier transform of h. After the injection we then whiten the data using the PSD re-estimated including the injected signal. The length of the data in this process is 16 seconds and we keep only the central 8 seconds to avoid FFT artifacts. We also purposely arrange for the signals to peak within the last 2 seconds of this final 8 seconds, in a hope that our model could detect the coalescence event as soon as it comes into the analyzed data. The 2-second diversity in the signal position is expected to lead to a better sensitivity of our model in event position. Our tests did show that such arrangement for signals (to peak within the last 2 seconds) for training helped largely not only for a higher sensitivity of early detections but also for a more complete mapping of the waveform.

We then slice and standardize the whitened data into 15 time windows of 1 second, each with 0.5-second overlaps with its neighbors. Finally, we mark a time window with 1 if it contains a GW signal of longer than 0.25 seconds, or longer than half of the signal duration when the duration is shorter than 0.5 second. All other windows are marked with 0. An example of the simulated data sample is shown in Fig. 2.

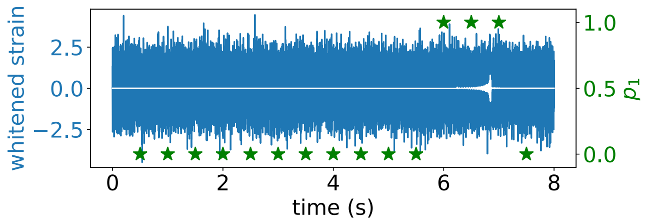

FIG. 2. Simulated LIGO L1 strain data (blue curve) that contain a GW signal (white curve) with $m_1 = 45.0$, $m_2 = 38.5$, $\rho_{\text{opt}} = 8$, right ascension $\pi/4$ and declination $\pi/4$. The green stars are the marks (0 or 1) of the time windows centered at each window.

To carry out this work, we generate 40960 and 4096 samples for the training dataset and the validation dataset respectively, each with half of the samples containing signals of SNRs randomly drawn between 5 and 15 in steps of 0.5. Similarly, for the testing dataset (used in Sec. III A to quantify the performance of our model), we generate 512 samples, half of which contain signals to cover an SNR range from 2 to 18 in steps of 0.5. For this testing performed in Sec. III A, we did not go for a larger dataset beyond the size of 512 because it took us about five days while already giving good quantitative results.

D. Training procedure

To deliver our model, we employ the tools provided by the software packages TensorFlow [83] and the TensorFlow Probability [84]. For the Bayesian LSTM layers we build a customized Bayesian LSTM cell that applies the weight perturbation and the flipout estimator on the hidden kernel, recurrent kernel and bias. The LSTM cell is then wrapped by the ordinary RNN layer functionality provided in TensorFlow. For updating the model weights during training, we use the ADAM [85] optimizer with a scheduled learning rate of

$$\text{lr}(e) = \begin{cases} 0.005, & \text{if } e \leq 15; \\ 0.005 * \exp(0.05 * (15 - e)), & \text{otherwise}; \end{cases} \tag{9}$$

where e is the number of training epoch. We use a combination of the sparse categorical cross entropy and the KL divergence as our loss function. Because in our training scheme the KL divergences of each layer and cell are accumulated into the regularization loss whenever passing through the data, we have to weigh the KL divergence in each layer and cell with the value of $\frac{1}{CN}$ [64], where C is the number of time steps in a sample and N is the total number of samples for training, so that it is applied only once per epoch. Otherwise the accumulated large KL divergence will overregularize the model and eventually stop the model from learning.

About the hardware, we train our model of 150 epochs and a batch size of 256 on a computer with Intel® Core™ i5 6500 CPU and AMD™ RX 480 GPU. Even with such moderate computational power and a code of Bayesian LSTM cell unoptimized for GPU, the whole training process normally takes only about 50 hours. This is a critical feature of our work. After the last epoch we save only the model parameters because we want to minimize the KL loss, which keeps decreasing during the training.

E. Flagging strategy

Our model performs tasks of binary classification, with a predictive value of p_1 indicating the identification for GW signals and a value of p_0 for noise, where $p_0 + p_1 = 1$. Thus we need to keep only p_1 for subsequent analysis. We also calculate the average predictive value $\bar{p}_1$ and the confidence score c_1 for $\bar{p}_1$ using Eqs. (4) and (7) respectively.

Considering the issue of confidence level in statistics, we weigh the $\bar{p}_1$ with its confidence score c_1. This is to avoid our model from accepting or rejecting a time window for signals when the confidence score c_1 is low. Thus we flag all the time windows with three distinct states: *trigger*, *noise*, and *awareness*. A time window t is flagged with a trigger state when the trigger score defined as

$$s_t = \bar{p}_{1,t} \times c_{1,t}, \tag{10}$$

is larger than 0.5, where $\bar{p}_{1,t}$ and $c_{1,t}$ are the $\bar{p}_1$ and c_1 of the time window t respectively. Such s_t can be regarded as the significance level of a detection.

Similarly a time window t is flagged with a noise state when the noise score defined as

$$\begin{aligned} n_t &= \bar{p}_{0,t} \times c_{0,t}, \\ &= (1 - \bar{p}_{1,t}) \times c_{1,t}, \end{aligned} \tag{11}$$

is larger than 0.5. We note that in a system of binary classification, $c_{0,t}$ equals $c_{1,t}$. We also note that $s_t + n_t = c_{1,t}$.

Those time windows that are not flagged with a trigger state nor a noise state will be flagged with an awareness state. The role of such a state can be illustrated in a Bayesian dog-null classifier for images when we feed it with an image of cat. It is likely that the classifier will give a reasonable p_1 but low c_1. In such a case we can only say that the model "notices" something though not sure what it is.

As a demonstration we apply the above flagging strategy to the data presented in Fig. 2 and the results are shown in Fig. 3. Because the flipout estimator enables batch prediction, we use a batch size of 32 to dramatically accelerate the MC sampling process (see Sec. IV B). As shown in Fig. 3, among the three time windows that are marked with 1 (green stars) for indicating the GW singles, our model flag two with trigger states (cyan stars) and one with an awareness state (cyan triangle) likely due to the much weaker signal within this time window. This demonstrates

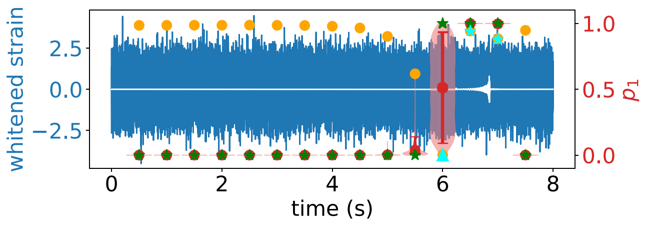

FIG. 3. The prediction with flagging results from our model for the input data presented in Fig. 2. The three cyan markers indicate the trigger scores s_t [Eq. (10)], where two trigger states are shown as stars and one awareness state as a triangle. We also show the confidence score $c_{1,t}$ (orange dots), averaged probability for being a signal $\bar{p}_{1,t}$ (red dots), distribution of $p_{1,t}$ (pink areas), and the 90% intervals of $p_{1,t}$ (red error bars), all based on a Monte-Carlo process with a sampling size of 4096.

the usefulness of the awareness state in practice, which has actually incorporated the sliding-window search. In Fig. 3 we also show the confidence score $c_{1,t}$ (orange dots), the averaged predictive probability for being a signal $\bar{p}_{1,t}$ (red dots), the distribution of $p_{1,t}$ (pink areas), and the 90% intervals of $p_{1,t}$ (red error bars), all obtained from a set of data sampled for 4096 times.

It is obvious that the awareness state is associated with a low confidence score $c_{1,t}$, which indicates a large uncertainty in a prediction. It suggests the need for further investigations on this particular time window. In practice, we could ignore these awareness states if we have had trigger states next to them because the purpose of our model is to detect signal events. However if the awareness states do not come with any neighboring trigger states, we could pursue further investigations in order not to miss any detection opportunities for signals of even unknown types. In a training process, on the other hand, these awareness states together with their known markers can be further incorporated into a retraining process, so that the retrained model could become more capable in discriminating between the signals and the noise rather than putting either into the awareness state.

III. RESULTS

A. Quantitative tests for performance

To quantify the performance of our model, we take four different measures, namely the TPR, TER, FPR, and FER as defined below, against the dataset generated in Sec. II C, which contains real noise and simulated signals. The MC sample size in obtaining $\bar{p}_1$ and c_1 is 4096. Because our model classifier gives three states instead of two, we cannot apply the commonly used confusion matrix to obtain the sensitivity or the false positive rate for our results.

The first measure is the "true positive rate" (TPR), which is the ratio of the trigger states plus the awareness states among all the time windows marked with "1" (with GW signals). To be precise, among all the time windows marked

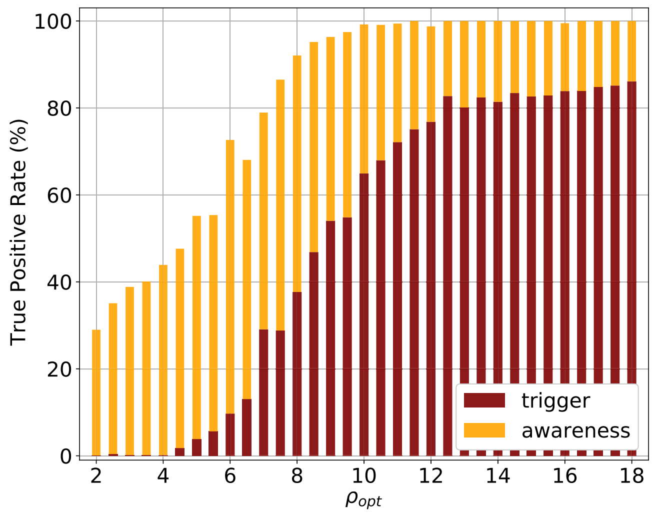

FIG. 4. The true positive rate (TPR) as a function of the SNR ρ_{opt} in the performance test of our model. The TPR reaches 90% when $\rho_{\mathrm{opt}} > 8$.

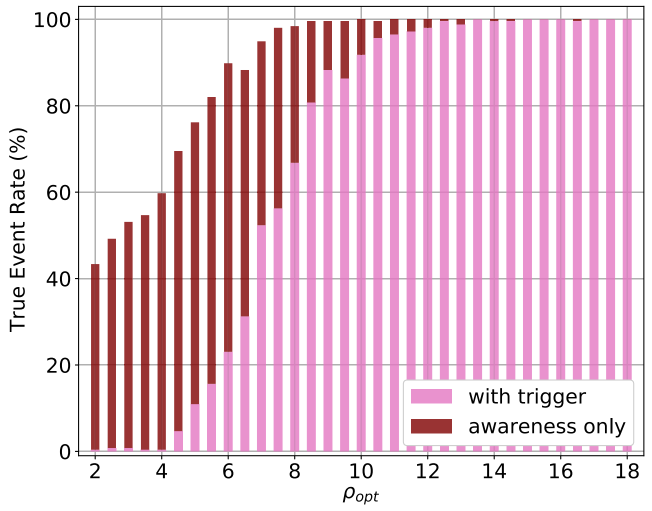

FIG. 5. The True Event Rate (TER) as a function of the SNR ρ_{opt} in the performance test of our model. The TER reaches 90% when $\rho_{\mathrm{opt}} > 7$.

with 1, if the numbers of states for trigger, awareness, and noise are N_{trg}, N_{aw}, and N_{noi} respectively, then the TPR is defined as

$$\mathrm{TPR} = \frac{N_{\mathrm{trg}} + N_{\mathrm{aw}}}{N_{\mathrm{trg}} + N_{\mathrm{aw}} + N_{\mathrm{noi}}}. \tag{12}$$

Such TPR can be regarded as the "waveform sensitivity," which indicates how well the model can capture the waveform structure. Figure 4 shows the results. It is clear that for GW signals with an SNR of $\rho_{\mathrm{opt}} > 8$, our model detects or is aware of more than 90% of the time windows in the GW waveforms. The persistent awareness rate of about 20% at high ρ_{opt} is due to the obscuration from the noise at the beginning parts of the waveforms, which are always weaker then the noise. In principle this rate of 20% could be reduced if we retrain the model with these marked awareness states.

The second measure is the "true event rate" (TER), which is the ratio of the events with at least one trigger or awareness among all the events with GW signals. Among all the signal events, if the numbers of events with "at least one trigger," with "awareness only," and with "noise only" are E_{trg}, E_{aw}, and E_{noi} respectively, then the TER is defined as

$$\mathrm{TER} = \frac{E_{\mathrm{trg}} + E_{\mathrm{aw}}}{E_{\mathrm{trg}} + E_{\mathrm{aw}} + E_{\mathrm{noi}}}. \tag{13}$$

Such TER can be regarded as the "event sensitivity," which indicates how well our model can pick up the true events. Fig. 5 shows the results. Our model achieves a TER of 90% when $\rho_{\mathrm{opt}} > 7$ and 100% when $\rho_{\mathrm{opt}} > 8.5$. For $\rho_{\mathrm{opt}} > 11$ we still see few cases with awareness only and our investigation shows that this is due to some outliers in the noise leading to low confidence.

In evaluating the TER and TPR results, it is quite encouraging to see the capability of our model in not only detecting the GW events (TER) but also identifying the full lengths of waveform durations in the events (TPR), although the latter demonstrates a slightly lower sensitivity. We also note that our model trained for the non-spinning BBHs successfully detected the spinning BBHs in similar statistics.

On the other hand, we could define two similar measures for false detections. Thus the third measure is the "false positive rate" (FPR), which is the ratio of the trigger states plus the awareness states among all the time windows marked with '0' (without GW signals). In other words, among all the time windows marked with 0, if the numbers of states for trigger, awareness, and noise are N^0_{trg}, N^0_{aw}, and N^0_{noi} respectively, then the FPR is defined as

$$\mathrm{FPR} = \frac{N^0_{\mathrm{trg}} + N^0_{\mathrm{aw}}}{N^0_{\mathrm{trg}} + N^0_{\mathrm{aw}} + N^0_{\mathrm{noi}}}, \tag{14}$$

which is the sum of the false trigger rate $N^0_{\mathrm{trg}}/(N^0_{\mathrm{trg}} + N^0_{\mathrm{aw}} + N^0_{\mathrm{noi}})$ and the false awareness rate $N^0_{\mathrm{aw}}/(N^0_{\mathrm{trg}} + N^0_{\mathrm{aw}} + N^0_{\mathrm{noi}})$. This FPR indicates how likely our model will falsely capture the waveform signals. Because the false detections are measured in a background of pure noise, the FPR is not a function of the SNR. Our model shows a false trigger rate of 0.067% and a false awareness rate of 19.6%, summed up to an FPR of about 19.7%. The false trigger rate is unnoticeably small and they are mainly due to the glitches in the noise. On the other hand, although the current false awareness rate of 19.6% is unignorable, we find that when conducting the same test using noise

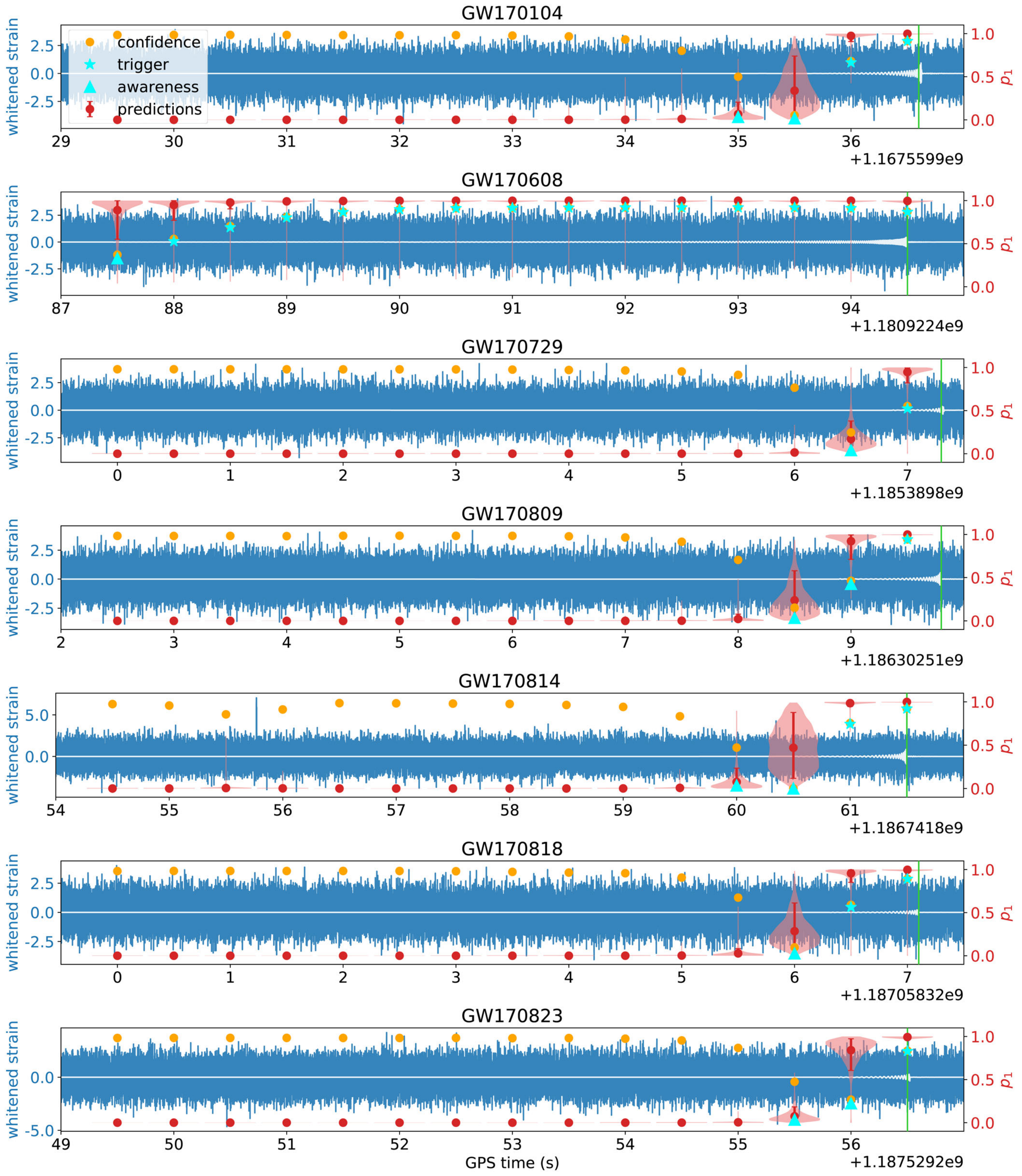

FIG. 6. Successful flagging results of our model for the LIGO Livingston O2 data that contain GW signals from BBH coalescence events. The green vertical lines indicate the event times. Other symbols and colors in these plots follow the same definitions as in Fig. 2 and Fig. 3.

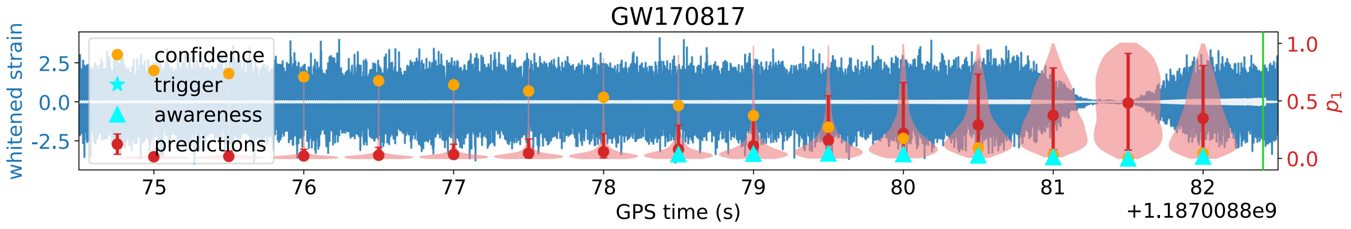

FIG. 7. The flagging results of the LIGO Livingston O2 data set GW170817, which contains a BNS coalescence event. Here we have applied an inverse Tukey window [13,87] to mitigate the glitch that occurred about 1.1 second before the event. The white curve is the reconstructed signal of the related BNS waveform.

datasets of different sizes for training, the false awareness rate decreases monotonically with the size of noise dataset for training. Therefore the current 19.6% is expected be further reduced by a larger noise training dataset, which would contain more variability in the background noise. In addition, we should be able to further reject false triggers or awareness when performing the coincidence test (see Sec. IV A).

The last measure is the "false event rate" (FER), which is the ratio of the events with at least one trigger or awareness among all the null events without GW signals. Among all the null events, if the numbers of events with at least one trigger, with awareness only, and with noise only are E^0_{trg}, E^0_{aw}, and E^0_{noi} respectively, then the FER is defined as

$$\mathrm{FER} = \frac{E^0_{\mathrm{trg}} + E^0_{\mathrm{aw}}}{E^0_{\mathrm{trg}} + E^0_{\mathrm{aw}} + E^0_{\mathrm{noi}}}, \tag{15}$$

which is the sum of the false event trigger rate $E^0_{\mathrm{trg}}/(E^0_{\mathrm{trg}} + E^0_{\mathrm{aw}} + E^0_{\mathrm{noi}})$ and the false event awareness rate $E^0_{\mathrm{aw}}/(E^0_{\mathrm{trg}} + E^0_{\mathrm{aw}} + E^0_{\mathrm{noi}})$. Our model shows a false event trigger rate of 0.485% and a false event awareness rate of 47.2%, summed up to an FER of about 47.7%. This shows that while our model is capable of detecting the true events as previously seen, it falsely overpredicts the signal events. Again this is simply due to the smallness of our noise dataset for training and can be improved by enlarging the noise training dataset as well as going deeper in the training.

B. Performance against real events

In this section we test our model with the LIGO Livingston O2 data that contain confidence detections of BBH coalescence events [6–9,86].

We downloaded the 32-second strain data from GWOSC, down-sampled them to 8192 Hz, calculated the PSD, whitened the data, and then took the 8-second chunks that contain the events in their last seconds. As in the training process, each chunk of the real data here was also sliced into 15 one-second windows with a stride of 0.5 second. For each chunk of the event data, we performed an MC sampling of size 4096, with a batch size of 32. When operated on a moderate GPU-equipped PC, the whole process of flagging took only about 20 seconds for each chunk. Thus our model should be able to achieve nearly real-time detection if we employ better hardware and optimize the Bayesian LSTM layer in gaining the full GPU support. Shorter data chunks and a smaller size of the MC sampling should also help on this.

The results for the BBH coalescence events are shown in Fig. 6. The symbols and colors in these plots follow the same definitions as in Fig. 2 and Fig. 3. For reference purpose, we also plot the whitened GW waveforms (white curves) reconstructed from the values of component masses, luminosity distances, right ascensions and declinations provided in Ref. [9] using the SEOBNRv4T model.

It is clear that our model successfully detected all the events. It is also important to note that our model successfully triggered all the time windows that contained the GW170608 signal, which spanned a period of nearly 7 seconds. This demonstrates the capability of our model in capturing the full length of a long-duration GW signal. While in literature the matched-filtering search detected GW170729 with rather high false alarm rate [9,86], our detection of GW170729 also comes with a relatively lower confidence score. This is likely due to the noise fluctuation in the background and could be improved by deepening the model or increasing the size of the training dataset.

Although our model was trained with only the BBH coalescence events, we experimentally tested our model against a binary-neutron-star (BNS) coalescence event, the GW170817 [13], which has masses of 1.27 $M_{\odot}$ and 1.46 $M_{\odot}$. Figure 7 shows the result. For this particular set of Livingston O2 data where there was a large glitch about 1.1 seconds before the event [13], we employed a method used for the rapid reanalysis in Ref. [13,87] to mitigate this large glitch. The method applies an inverse Tukey window to zero out the data around the glitch. For reference purpose we also plotted the reconstructed BNS GW waveform (white curve) using the TaylorT4 model [88]. As shown in Fig. 7, our model was aware (with no triggers) of the time windows that contained the waveform of the BNS event.

To quantify how much our result had been affected by the Tukey window, which actually deformed the strain data

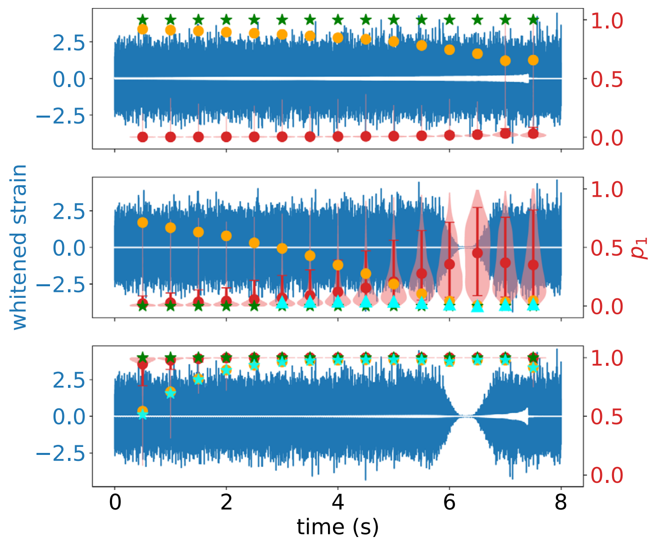

FIG. 8. Three runs for cross validation to test the influence of a Tukey window on the GW170817 result in Fig. 7. With the same normal noise background from the LIGO data, the first run contained a BNS waveform same as shown in Fig. 7 to mimic the GW170817, without any window treatment (top panel); the second run contained no waveform but applied with an inverse Tukey window (middle panel); the third contained a BBH waveform to mimic a loud BBH event, with an inverse Tukey window (bottom panel).

and can be thought as an adversarial example [89,90], we performed the following three runs for cross validation (see Fig. 8). With the same normal noise background (without glitches) from the LIGO Livingston O2 data, the first run contained a BNS waveform same as shown in Fig. 7 to mimic the GW170817, without any window treatment (top panel in Fig. 8); the second run contained no waveform but applied with an inverse Tukey window (middle panel in Fig. 8); the third contained a BBH waveform to mimic a loud BBH event ($m_1 = 8.5\ M_\odot$, $m_2 = 6.0\ M_\odot$, and $\rho_{\rm opt} = 12$), with an inverse Tukey window (bottom panel in Fig. 8). It is evident that our model rejected the GW170817-like signal in the top panel, raised awareness for the Tukey window in the middle panel, and raised triggers for the BBH event in the bottom panel without being affected by the Tukey window. We also note that the $\bar{p}_1$ in the middle panel exhibits a behavior similar to the one shown in Fig. 7. Therefore it is likely that the awareness seen in the GW170817 (Fig. 7) is due to the treatment involving the Tukey window.

We further investigated the sensitivity of our model for the BNS events. Figure 9 shows an example where the BNS signal is as loud as $\rho_{\rm opt} = 40$, considered as a nearby event closer to the observer. In this case our model was aware of most time windows containing the waveform and gave low $\bar{p}_1$ with significant confidence scores near the coalescence event. This means that our model that was trained with the BBH events (of total component mass larger than $10\ M_\odot$)

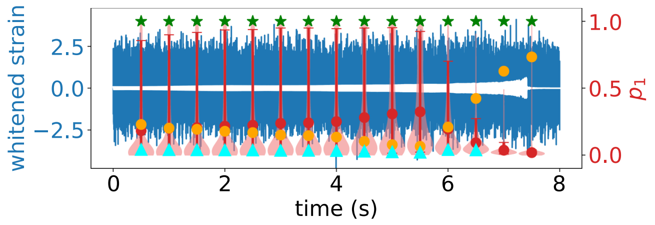

FIG. 9. The flagging result of a loud BNS event with $\rho_{\rm opt} = 40$.

is insensitive to the BNS events. This is not too surprising because the BNS events involve a much smaller mass range and could have quite different features in their GW waveforms as compared with the BBH events. Therefore to possess sensitivity for the BNS events, we would need to add the BNS waveform templates into our training dataset so that our model could recognize it as a separate BNS class. Although this process is straightforward in our framework (see Sec. IV D), we did not include the BNS training in this demonstrative work due to its large variation in waveform templates thus requiring much more computation power than we had.

In summary, our model is capable of detecting the BBH events, unaffected by the treatment involving the Tukey windowing. Our current model trained with only the BBH events will not falsely detect the BNS events but could potentially raise awareness for them. In the near future a multiclass classifier will be needed for the forthcoming new GW data. We have more discussions on this in Sec. IV D.

IV. DISCUSSION

In this section we discuss various critical issues in our model, further demonstrating its strengths and future potentials.

A. Extensibility for multidetector network

In contemporary GW searches, multidetector network is becoming crucial so the coincidence test [30,31,33,91,92] has become critical for declaring detections. The coincidence test is based on the fact that all detectors should have observed the same event at the same time so that the triggers due to artifacts can be minimized by cross-correlating the data from different detectors. Our model is readily extensible for this as described below.

To make a neural network model capable of performing coincidence test, we need to find a way to feed the multidetector data into the model so that the convolutional layers can perform the cross-correlation to extract useful information from the data [93]. Fortunately our model can readily do that by simply stacking the multidetector data to form a multichannel array as its input. Although this process is rather straightforward for our current setup, the size of training dataset will grow in proportion to the number of detectors, thus requiring proportionally more computation power.

We also note that a potential advantage of the coincidence test based on the multidetector data is the management of noise glitches. For data from a signal detector we normally need more thorough feeding of experimental glitches into the model so that our model could learn to ignore the glitches when trying to detect signals. This requires much more computation power than what we have performed. On the other hand a model including the coincidence test for multidetector data should be able to serve the same, without the need for the model to learn all the characteristics of possible noise. We plan to demonstrate the coincidence test in our future works.

B. Size of Monte-Carlo sampling

The first is about the trade-off between efficiency and accuracy in our Bayesian neural network. In theory the accuracy of BNN increases monotonically with the size of Monte-Carlo sampling, which is the main determinant of the required computation time. Thus there is no natural way for achieving both high accuracy and high efficiency. For example a smaller sampling size helps speed up the prediction process while on the other hand inducing larger uncertainty in the predicted $\bar{p}_1$ and c_1. To find a proper trade-off we quantified the dependence of both the prediction uncertainty and the computation time separately on the MC sampling size and also on the batch size for the flipout estimator, which enables parallel predictions (see Sec. II).

About MC sampling, we considered the sizes of 128, 256, 512, 1024, 2048, 4096, 8192, and 16384, each conducted for 100 times with the data in Fig. 2 to obtain the averaged variances in $\bar{p}_1$ and c_1 and the average computation time. The batch size for the flipout estimator is 32 throughout this test. The left panels in Fig. 10 show the results. As expected, while both the variances in $\bar{p}_1$ and c_1 decrease with the sampling size, the computation time increases linearly. Based on these results, we chose 4096 as a reasonable trade-off size for the MC sampling.

About the batch size for the flipout estimator, we considered the sizes of 2, 4, 8, 16, 32, 64, 128, 256, and 512, all with the same total sampling size of 4096 and conducted for 100 times using the data in Fig. 2. This test is to understand whether or not a larger batch size does improve much on the computation time, because a larger batch size actually comes with a possible price of sampling bias. The right panels in Fig. 10 show the results. It is clear that the gain in computation efficiency follows the expected behavior of linearity but only for batch sizes up to about 32 but then becomes marginal for larger batch sizes.

For the above reasons, we chose to perform all the predictions in this work (as in Sec. III) with the sampling size of 4096 and the batch size of 32. In principle, one should increase the sampling size while keeping the batch size small whenever practically feasible.

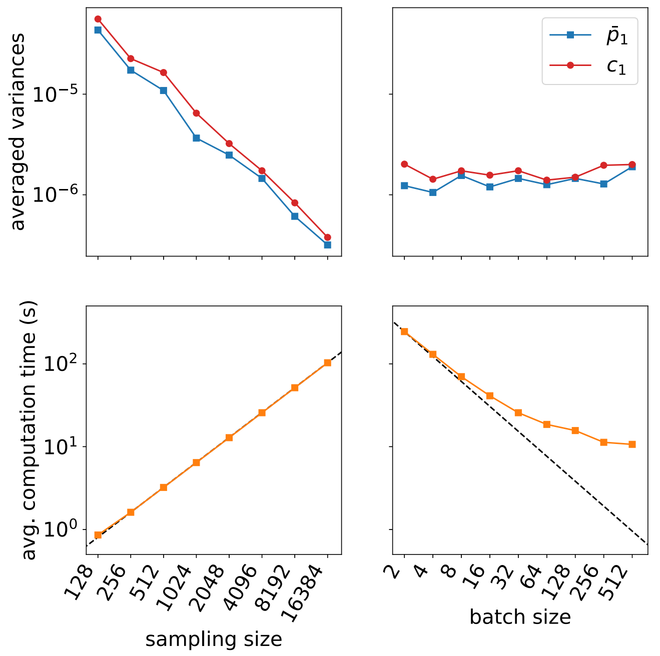

FIG. 10. The left panels show the dependence of averaged variances in $\bar{p}_1$ and c_1 (top) and of the average computation time (bottom) on the MC sampling size. The black dashed line in the bottom-left panel represents the linearity of slope 1. The right panels show the dependence of averaged variances in $\bar{p}_1$ and c_1 (top) and of the average computation time (bottom) on the batch size for the flipout estimator, all with an MC sampling size of 4096. The black dashed line in the bottom-right panel represents the inverse linearity of slope −1. All results are based on the data presented in Fig. 2.

C. Model calibration and reliability

It has been suggested that the confidence of the predictions from a neural network classifier may be justified directly via its predictive values. That is to compare its predictive values with the empirical frequencies of the input data [94–96]. If they match well, this neural network classifier is said to have been well-calibrated and thus possesses confidence in its reliability. To this end, we first followed the method in Ref. [94] to produce the reliability curve, and then computed the expected calibration error (ECE) as defined in Ref. [95], using our validation dataset generated in Sec. II C 3. Each of the 4096 samples in our validation dataset possesses 15 time windows, each with a predictive value $\bar{p}_1$. To construct the reliability curve, all these 4096×15 time windows were sorted in their associated $\bar{p}_1$ and then divided into 10 bins of roughly equal size. Within each bin, the $\bar{p}_1$'s were averaged to become the horizontal coordinate in the left panel of Fig. 11, while the actual fraction of real events (the positives) formed the vertical coordinate. The 10 bins thus generated 10 points in the plot, linked to form the reliability curve (the red solid line). Ideally we would like this curve to be as close as possible to the perfect curve, which has a slope of one

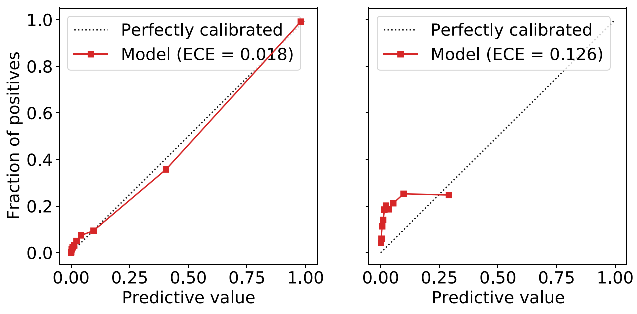

FIG. 11. The reliability curves of our model based on the validation dataset (left) and on only a subset with a low SNR of $\rho_{\rm opt} = 2.5$ (right).

through the origin (the dotted line). It is clear that our model is fairly well-calibrated with a very small error of ECE $= 0.018$. We could thus conclude that the $\bar{p}_1$ generated by our model is a confident estimate for reality.

However, such confidence may break down in the following two cases. The first is when the input data does not follow the same distribution as the training dataset. The fact that our model, currently trained with only the BBH events, is insensitive to the BNS events is one example (see Sec. III B). This is why we introduced and utilized the confidence score c_k (see Eq. (7)) in our work. The confidence score would be low whenever the input data do not comply with the underlying distribution of the training data. This out-of-distribution problem can be minimized by training our model with datasets of all possible kinds that we may encounter during the observation.

The second case for the confidence breakdown is when the signal-to-noise radio is not high enough so that the model fails to detect all the features it is supposed to recognize. To demonstrate this point, we computed the reliability curve for a subset of our validation data with $\rho_{\rm opt} = 2.5$. The right panel of Fig. 11 shows the result. The reliability curve (red solid) being well to the left of the perfect curve (dotted) with a significant error of ECE $= 0.126$ indicates under detection of real events (the positives). This is simply due to the obscuration of the GW signals by the noise. Such a problem can be readily improved by deepening our model training to gain higher sensitivity.

To sum up, we may train our model deeper to gain higher sensitivity and also feed it with more signal types to avoid the breakdown of confidence. Furthermore, as long as our model possesses high enough sensitivity through deep training, our uncertainty estimation based on the confidence score could serve as a powerful tool for picking up the out-of-distribution signals that we have never seen before.

D. Multiclass detection

In this work we only trained our model with BBH signals to perform binary classification. However, as more BNS signals have been discovered by LIGO and Virgo [13,14], a classifier that can further distinguish among different types of signals such as those from BBH, BNS, and BHNS is needed. For example the deterministic CNN proposed in Ref. [44] is able to distinguish between the BBH and BNS signals. A multiclass classifier can not only reduce the search space of matched-filtering but also improve the uncertainty estimation in the Bayesian approach that we propose here. To estimate the uncertainties in signal predictions for a multiclass Bayesian approach, we need to employ the covariance matrix defined in Eq. (5), which is more complicated than Eq. (6) used in this work. In turn the definition of the confidence score in Eq. (7) also needs to be updated accordingly. We will leave these to future works.

E. Stride size of sliding window

The architecture of our model combines the CNN model and the sliding-window searches. Here we illustrate the effect of the stride size of our sliding windows. In principle, while keeping the same window size, a smaller stride size will increase the time resolution in pinning down a GW event. Figure 12 shows the flagging results of the LIGO Livingston GW170809 data using stride sizes of 0.25 (top) and 0.7 (bottom) second, to be compared with our previous result for using 0.5 second (the fourth panel in Fig. 6). All time windows have the same size of one second. It is evident that the 0.25-second result demonstrates the best accuracy in labeling the temporal location of the GW event.

The only drawback for a smaller stride size is that the computation time is inversely increasing with the stride size, so there is always a trade-off given the available computation power. Another issue concerning the stride size is that a stride size larger than half of the window size will cause an un-uniform coverage of the time domain in the flagging analysis, leading to possible misses for the GW events [41]. Therefore it is optimal to choose a stride size of half the window size in face of the always limited computation power. In the main work presented in this paper we have chosen a window size of one second with a stride size of 0.5 second. When analyzing the forthcoming new data in future, one can always use our model here to first efficiently identify a GW event, hopefully in nearly real time, and then focus on the identified data chunks with increased time resolution and deeper search or even further using the matched-filtering search to pinpoint the coalescence time.

F. Forecasting GW events

It is always desirable to be able to detect the GW waves before a coalescence event so that we could conduct parallel targeted observations such as the EM observations to trace the full process including the inspiral, merger and ringdown phases. A similar desire in the trading market has actually led to using the RNN model to perform time series forecasting in detecting the anomaly in time series [97] and in predicting the stock price [98]. However our techniques based on deep learning for detecting GW waves rely

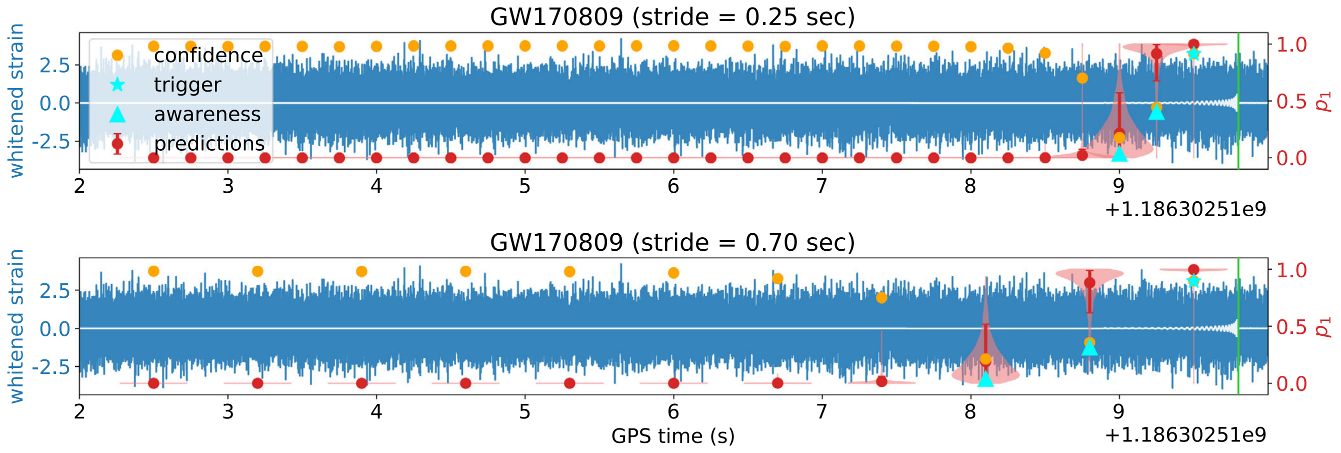

FIG. 12. The flagging results of the Livingston GW170809 data using sliding-window search with stride sizes of 0.25 second (top) and 0.7 second (bottom).

strongly on recognizing the features of the GW waveform around the coalescence time, where the SNR is much stronger. This in turn implies that our model could detect the GW signals likely only when a coalescence event has come into the scene, and then we trace back in time to identify the data sections that may have contained the GW waveform before the event.

We verify this here using the Livingston GW170608 data. The results are shown in Fig. 13, where the bottom panel contains only one extra second where the coalescence event sits in its middle. The top panel shows no sign of GW detection even though the event is about to enter our time domain of analysis, only half a second away. Once the event enters our time domain (bottom panel), our model immediately picks up all the time windows containing not only the event but also the waveform prior to the event. This is not surprising because for each forward pass our model uses the zero states as the initial hidden and cell states.

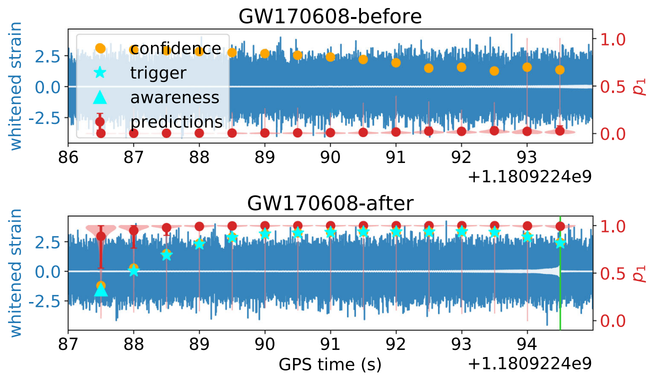

FIG. 13. The flagging results on the Livingston GW170608 data without the coalescence event (top) and with the event (bottom). The only difference between these two sub-datasets is the inclusion of the extra one second (on the right) in the bottom panel where the event sits in the middle of this second.

To enable the capability for prior detections, we have to incorporate the hidden and cell states of the previous prediction into our model. This would require not only modifications in our model structure but also the construction of partly correlated samples in the training dataset. An extra complication is that the Bayesian neural network would need to have all the previous hidden and cell states in the form of distributions. Considering these challenges we will leave the attempts for forecasting to future work.

V. CONCLUSION

We have proposed a new model to demonstrate the capability of Bayesian neural network in detecting gravitational waves. One critical bonus for using the BNN is the ability for uncertainty estimation, which is particularly useful when facing the data that do not follow the distribution of the training dataset. Our uncertainty estimation is manifested by the newly defined confidence score c_1, which in turn defines the awareness state for collecting the cases where triggers cannot be accepted nor rejected with confidence. These cases can then be further investigated in the follow-up checks. In addition, our Bayesian CLDNN model integrates the CNN classifier with the sliding-window search scheme so that we could detect most of the time windows that contain the GW waveform in a coalescence event. However due to the limited computation power of this work, our current model does not outperform the sensitivity of the existing matched-filtering search but still successfully detect all the GW events that LIGO detected in the O2 observation. Nevertheless on a moderate GPU-equipped personal computer, it takes only about 20 seconds for detecting an event and labeling its waveform period whenever a coalescence event comes into the time domain of our analysis. This 20-second latency is expected to be dramatically improved by a GPU-optimized code with enhanced computation power and even shorter data chunks, making our model possible for nearly real-time detection. Such nearly real-time detection is unlikely to be possible for the matched-filtering search, which is always computationally expensive. In future we plan to further explore the discussed potential for a Bayesian CLDNN model in forecasting the GW coalescence events.

ACKNOWLEDGMENTS

We would like to thank Chun Hsiang Chan for useful discussions on deep learning and statistics. This research has used the data, software and/or web tools from the Gravitational Wave Open Science Center [99], a service of LIGO Laboratory, the LIGO Scientific Collaboration and the Virgo Collaboration. LIGO is funded by the U.S. National Science Foundation. Virgo is funded by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale della Fisica Nucleare (INFN) and the Dutch Nikhef, with contributions by Polish and Hungarian institutes. We acknowledge the support from Ministry of Science and Technology, Taiwan.

[1] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Observation of Gravitational Waves from a Binary Black Hole Merger, Phys. Rev. Lett. **116**, 061102 (2016).
[2] B. P. Abbott *et al.*, Advanced LIGO, Classical Quantum Gravity **32**, 074001 (2015).
[3] F. Acernese *et al.*, Advanced Virgo: A second-generation interferometric gravitational wave detector, Classical Quantum Gravity **32**, 024001 (2015).
[4] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), GW151226: Observation of Gravitational Waves from a 22-Solar-Mass Binary Black Hole Coalescence, Phys. Rev. Lett. **116**, 241103 (2016).
[5] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), Binary Black Hole Mergers in the First Advanced LIGO Observing Run, Phys. Rev. X **6**, 041015 (2016).
[6] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), GW170104: Observation of a 50-Solar-Mass Binary Black Hole Coalescence at Redshift 0.2, Phys. Rev. Lett. **118**, 221101 (2017).
[7] B. P. Abbott *et al.*, GW170608: Observation of a 19 solar-mass binary black hole coalescence, Astrophys. J. **851**, L35 (2017).
[8] B. P. Abbott *et al.* (LIGO Scientific Collaboration and Virgo Collaboration), GW170814: A Three-Detector Observation of Gravitational Waves from a Binary Black Hole Coalescence, Phys. Rev. Lett. **119**, 141101 (2017).
[9] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaboration), GWTC-1: A Gravitational-Wave Transient Catalog of Compact Binary Mergers Observed by LIGO and Virgo during the First and Second Observing Runs, Phys. Rev. X **9**, 031040 (2019).
[10] R. Abbott *et al.* (LIGO Scientific, Virgo Collaboration), GW190521: A Binary Black Hole Merger with a Total Mass of 150 $M_{\odot}$, Phys. Rev. Lett. **125**, 101102 (2020).
[11] R. Abbott *et al.* (LIGO Scientific, Virgo Collaboration), GW190814: Gravitational waves from the coalescence of a 23 solar mass black hole with a 2.6 solar mass compact object, Astrophys. J. Lett. **896**, L44 (2020).
[12] R. Abbott *et al.* (LIGO Scientific, Virgo Collaboration), GW190412: Observation of a binary-black-hole coalescence with asymmetric masses, Phys. Rev. D **102**, 043015 (2020).
[13] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral, Phys. Rev. Lett. **119**, 161101 (2017).
[14] B. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), GW190425: Observation of a compact binary coalescence with total mass $\sim 3.4\ M_{\odot}$, Astrophys. J. Lett. **892**, L3 (2020).
[15] A. Goldstein *et al.*, An ordinary short gamma-ray burst with extraordinary implications:fermi-GBM detection of GRB 170817a, Astrophys. J. **848**, L14 (2017).
[16] V. Savchenko *et al.*, INTEGRAL Detection of the first prompt gamma-ray signal coincident with the gravitational-wave event GW170817, Astrophys. J. **848**, L15 (2017).
[17] B. P. Abbott *et al.*, Gravitational waves and gamma-rays from a binary neutron star merger: GW170817 and GRB 170817a, Astrophys. J. **848**, L13 (2017).
[18] M. Soares-Santos *et al.* (DES, LIGO Scientific, Virgo Collaborations), First measurement of the hubble constant from a dark standard siren using the dark energy survey galaxies and the LIGO/Virgo binaryblack-hole merger GW170814, Astrophys. J. **876**, L7 (2019).
[19] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Constraints on cosmic strings using data from the first Advanced LIGO observing run, Phys. Rev. D **97**, 102002 (2018).
[20] T. Akutsu *et al.* (KAGRA Collaboration), KAGRA: 2.5 Generation interferometric gravitational wave detector, Nat. Astron. **3**, 35 (2019).
[21] B. Iyer, T. Souradeep, C. Unnikrishnan, S. Dhurandhar, S. Raja, and A. Sengupta, LIGO-India, proposal of the consortium for Indian initiative in gravitational-wave observations (indigo), Report No. lIGO-M1100296-v2, 2011.
[22] B. P. Abbott *et al.*, Prospects for observing and localizing gravitational-wave transients with Advanced LIGO, Advanced Virgo and KAGRA, Living Rev. Relativity **21**, 3 (2018).
[23] B. D. Metzger and E. Berger, What is the most promising electromagnetic counterpart of A neutron star binary merger? Astrophys. J. **746**, 48 (2012).

[24] M. L. Chan, Y.-M. Hu, C. Messenger, M. Hendry, and I. S. Heng, Maximizing the detection probability of kilonovae associated with gravitational wave observations, Astrophys. J. **834**, 84 (2017).

[25] L. Blackburn, M. S. Briggs, J. Camp, N. Christensen, V. Connaughton, P. Jenke, R. A. Remillard, and J. Veitch, High-energy electromagnetic offline follow-up of LIGO-Virgo gravitational-wave binary coalescence candidate events, Astrophys. J. Suppl. Ser. **217**, 8 (2015).

[26] C. Pankow, L. Sampson, L. Perri, E. Chase, S. Coughlin, M. Zevin, and V. Kalogera, Astrophysical prior information and gravitational-wave parameter estimation, Astrophys. J. **834**, 154 (2017).

[27] M. W. Coughlin *et al.*, Optimizing searches for electromagnetic counterparts of gravitational wave triggers, Mon. Not. R. Astron. Soc. **478**, 692 (2018).

[28] B. J. Owen and B. S. Sathyaprakash, Matched filtering of gravitational waves from inspiraling compact binaries: Computational cost and template placement, Phys. Rev. D **60**, 022002 (1999).

[29] A. Nitz *et al.*, gwastro/pycbc: PyCBC release v1.15.2 (2019).

[30] S. A. Usman *et al.*, The PyCBC search for gravitational waves from compact binary coalescence, Classical Quantum Gravity **33**, 215004 (2016).

[31] S. Sachdev *et al.*, The GstLAL search analysis methods for compact binary mergers in Advanced LIGO's second and Advanced Virgo's first observing runs, arXiv:1901.08580.

[32] C. Messick *et al.*, Analysis framework for the prompt discovery of compact binary mergers in gravitational-wave data, Phys. Rev. D **95**, 042001 (2017).

[33] K. Cannon *et al.*, Toward early-warning detection of gravitational waves from compact binary coalescence, Astrophys. J. **748**, 136 (2012).

[34] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1991).

[35] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. **1**, 541 (1989).

[36] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).

[37] H. Sak, A. Senior, and F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (International Speech Communication Association (ISCA), Singapore, 2014).

[38] F. A. Gers, J. Schmidhuber, and F. Cummins, Learning to forget: Continual prediction with lstm, Neural Comput. **12**, 2451 (2000).

[39] D. George and E. A. Huerta, Deep neural networks to enable real-time multimessenger astrophysics, Phys. Rev. D **97**, 044039 (2018).

[40] D. George and E. A. Huerta, Deep learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data, Phys. Lett. B **778**, 64 (2018).

[41] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schlkopf, Convolutional neural networks: A magic bullet for gravitational-wave detection? Phys. Rev. D **100**, 063015 (2019).

[42] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Matching Matched Filtering with Deep Networks for Gravitational-Wave Astronomy, Phys. Rev. Lett. **120**, 141103 (2018).

[43] X. Fan, J. Li, X. Li, Y. Zhong, and J. Cao, Applying deep neural networks to the detection and space parameter estimation of compact binary coalescence with a network of gravitational wave detectors, Sci. China Phys. Mech. Astron. **62**, 969512 (2019).

[44] P. G. Krastev, Real-time detection of gravitational waves from binary neutron stars using artificial neural networks, Phys. Lett. B **803**, 135330 (2020).

[45] B. Lin, X. Li, and W. Yu, Binary neutron stars gravitational wave detection based on wavelet packet analysis and convolutional neural networks, arXiv:1910.10525.

[46] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy, arXiv:1909.06296.

[47] M. Zevin *et al.*, Gravity Spy: Integrating Advanced LIGO detector characterization, machine learning, and citizen science, Classical Quantum Gravity **34**, 064003 (2017).

[48] D. George, H. Shen, and E. A. Huerta, Classification and unsupervised clustering of LIGO data with Deep Transfer Learning, Phys. Rev. D **97**, 101501(R) (2018).

[49] M. Llorens-Monteagudo, A. Torres-Forn, J. A. Font, and A. Marquina, Classification of gravitational-wave glitches via dictionary learning, Classical Quantum Gravity **36**, 075005 (2019).

[50] S. Coughlin, S. Bahaadini, N. Rohani, M. Zevin, O. Patane, M. Harandi *et al.*, Classifying the unknown: Discovering novel gravitational-wave detector glitches using similarity learning, Phys. Rev. D **99**, 082002 (2019).

[51] H. Shen, D. George, E. A. Huerta, and Z. Zhao, Denoising gravitational waves with enhanced deep recurrent denoising auto-encoders, arXiv:1903.03105.

[52] W. Wei and E. A. Huerta, Gravitational wave denoising of binary black hole mergers with deep learning, Phys. Lett. B **800**, 135081 (2020).

[53] H. Shen, D. George, E. A. Huerta, and Z. Zhao, Denoising gravitational waves using deep learning with recurrent denoising autoencoders, arXiv:1711.09919.

[54] R. Ormiston, T. Nguyen, M. Coughlin, R. X. Adhikari, and E. Katsavounidis, Noise reduction in gravitational-wave data via deep learning, Phys. Rev. Research **2**, 033066 (2020).

[55] A. Nguyen, J. Yosinski, and J. Clune, Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE Computer Society, Boston, MA, 2015), https://doi.org/10.1109/CVPR.2015.7298640.

[56] D. J. C. MacKay, A practical Bayesian framework for backpropagation networks, Neural Comput. **4**, 448 (1992).

[57] R. M. Neal, Bayesian learning for neural networks, Ph.D. thesis, University of Toronto, 1995.

[58] Y. Gal and Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, arXiv:1506.02142.

[59] A. Graves, Practical variational inference for neural networks, in *Proceedings of the 24th International Conference*

on Neural Information Processing Systems, NIPS'11 (Curran Associates Inc., USA, 2011), pp. 2348–2356.
[60] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, Weight uncertainty in neural networks, arXiv:1505.05424.
[61] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, Flipout: Efficient pseudo-independent weight perturbations on mini-batches, arXiv:1803.04386.
[62] D. P. Kingma and M. Welling, Auto-encoding variational bayes, arXiv:1312.6114.
[63] D. P. Kingma, T. Salimans, and M. Welling, Variational dropout and the local reparameterization trick, arXiv:1506.02557.
[64] M. Fortunato, C. Blundell, and O. Vinyals, Bayesian recurrent neural networks, arXiv:1704.02798.
[65] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, Convolutional, long short-term memory, fully connected deep neural networks, in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE Computer Society, South Brisbane, QLD, 2015), pp. 4580–4584, https://doi.org/10.1109/ICASSP.2015.7178838.
[66] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, An introduction to variational methods for graphical models, Mach. Learn. **37**, 183 (1999).
[67] S. Kullback and R. A. Leibler, On information and sufficiency, Ann. Math. Stat. **22**, 79 (1951).
[68] S. Kullback, *Information Theory and Statistics* (Courier Corporation, New York, USA, 1997).
[69] A. Kendall and Y. Gal, What uncertainties do we need in bayesian deep learning for computer vision?, in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., Long Beach, CA, 2017), pp. 5574–5584.
[70] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation, Comput. Stat. Data Anal. **142**, 106816 (2020).
[71] G. Deodato, C. Ball, and X. Zhang, Bayesian neural networks for cellular image classification and uncertainty analysis (2020), https://doi.org/10.1101/824862.
[72] T. Gebhard, N. Kilbertus, G. Parascandolo, I. Harry, and B. Schölkopf, Convwave: Searching for gravitational waves with fully convolutional neural nets, in *Workshop on Deep Learning for Physical Sciences (DLPS) at the 31st Conference on Neural Information Processing Systems* (Neural Information Processing Systems Foundation (NeurlIPS), Long Beach, CA, 2017).
[73] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, WaveNet: A generative model for raw audio, arXiv:1609.03499.
[74] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).
[75] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016), http://www.deeplearningbook.org.
[76] M. Riesenhuber and T. Poggio, Hierarchical models of object recognition in cortex, Nat. Neurosci. **2**, 1019 (1999).
[77] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv:1502.03167.
[78] V. Nair and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in *the 27th International Conference on Machine Learning, Haifa, Israel, 2010* (ICML, 2010).
[79] M. Vallisneri, J. Kanner, R. Williams, A. Weinstein, and B. Stephens, The LIGO open science center, J. Phys. Conf. Ser. **610**, 012021 (2015).
[80] LIGO Scientific Collaboration, LIGO Algorithm Library—LALSuite, free software (GPL), https://doi.org/10.7935/GT1W-FZ16 (2018).
[81] A. Bohe, L. Shao, A. Taracchini, A. Buonanno, S. Babak, I. W. Harry *et al.*, Improved effective-one-body model of spinning, nonprecessing binary black holes for the era of gravitational-wave astrophysics with advanced detectors, Phys. Rev. D **95**, 044028 (2017).
[82] B. S. Sathyaprakash and B. F. Schutz, Physics, astrophysics and cosmology with gravitational waves, Living Rev. Relativity **12**, 2 (2009).
[83] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from www.tensorflow.org.
[84] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, TensorFlow distributions, arXiv:1711.10604.
[85] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.
[86] B. P. Abbott *et al.*, Low-latency gravitational-wave alerts for multimessenger astronomy during the second Advanced LIGO and Virgo observing run, Astrophys. J. **875**, 161 (2019).
[87] C. Pankow *et al.*, Mitigation of the instrumental noise transient in gravitational-wave data surrounding GW170817, Phys. Rev. D **98**, 084016 (2018).
[88] M. Boyle, D. A. Brown, L. E. Kidder, A. H. Mroue, H. P. Pfeiffer, M. A. Scheel, G. B. Cook, and S. A. Teukolsky, High-accuracy comparison of numerical relativity simulations with post-Newtonian expansions, Phys. Rev. D **76**, 124038 (2007).
[89] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, Intriguing properties of neural networks, arXiv:1312.6199.
[90] N. Akhtar and A. Mian, Threat of adversarial attacks on deep learning in computer vision: A survey, IEEE Access **6**, 14410 (2018).
[91] T. Adams, D. Buskulic, V. Germain, G. Guidi, F. Marion, M. Montani, B. Mours, F. Piergiovanni, and G. Wang, Low-latency analysis pipeline for compact binary coalescences in the advanced gravitational wave detector era, Classical Quantum Gravity **33**, 175012 (2016).
[92] S. Klimenko, I. Yakushin, A. Mercer, and G. Mitselmakher, A coherent method for detection of gravitational wave bursts, Classical Quantum Gravity **25**, 114029 (2008).
[93] L. E. Atlas, T. Homma, and R. J. M. II, An artificial neural network for spatio-temporal bipolar patterns: Application to phoneme classification, in *Neural Information Processing Systems*, edited by D. Z. Anderson (American Institute of Physics, Maryland, 1988), pp. 31–40.

[94] A. Niculescu-Mizil and R. Caruana, Predicting good probabilities with supervised learning, in *the 22nd International Conference on Machine Learning, Bonn, Germany, 2005* (ICML, 2005).

[95] M. Pakdaman Naeini, G. Cooper, and M. Hauskrecht, Obtaining well calibrated probabilities using bayesian binning, AAAI Conf. Artif. Intell. **2015**, 2901 (2015), https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4410090/.

[96] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, On calibration of modern neural networks, arXiv:1706.04599.

[97] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, Long short term memory networks for anomaly detection in time series, Presses universitaires de Louvain **89**, 89 (2015).

[98] A. Elliot and C. H. Hsu, Time series prediction: Predicting stock price, arXiv:1710.05751.

[99] https://www.gw-openscience.org.