Assessing the impact of non-Gaussian noise on convolutional neural networks that search for continuous gravitational waves

Takahiro S. Yamamoto,[1,*] Andrew L. Miller,[2,†] Magdalena Sieniawska,[2] and Takahiro Tanaka[3,4]

[1]*Department of Physics, Nagoya University, Nagoya 464-8602, Japan*
[2]*Université catholique de Louvain, Chemin du Cyclotron 2, B-1348 Louvain-la-Neuve, Belgium*
[3]*Graduate School of Science, Kyoto University, Kyoto 606-8502, Japan*
[4]*Center for Gravitational Physics, Yukawa Institute for Theoretical Physics, Kyoto University, Kyoto 606-8502, Japan*

(Received 3 June 2022; accepted 5 July 2022; published 18 July 2022)

We present a convolutional neural network that is capable of searching for continuous gravitational waves, quasimonochromatic, persistent signals arising from asymmetrically rotating neutron stars, in approximately one year of simulated data that is plagued by nonstationary, narrow-band disturbances, i.e., lines. Our network has learned to classify the input strain data into four categories: (1) only Gaussian noise, (2) an astrophysical signal injected into Gaussian noise, (3) a line embedded in Gaussian noise, and (4) an astrophysical signal contaminated by both Gaussian noise and line noise. In our algorithm, different frequencies are treated independently; therefore, our network is robust against sets of evenly spaced lines, i.e., combs, and we only need to consider a perfectly sinusoidal line in this work. We find that our neural network can distinguish between astrophysical signals and lines with high accuracy. In a frequency band without line noise, the sensitivity depth of our network is about $\mathcal{D}^{95\%} \simeq 43.9$ with a false alarm probability of ∼0.5%, while in the presence of line noise, we can maintain a false alarm probability of ∼10% and achieve $\mathcal{D}^{95\%} \simeq 3.62$ when the line noise amplitude is $h_0^{\mathrm{line}}/\sqrt{S_{\mathrm{n}}(f_k)} = 1.0$. The network is robust against the time derivative of the frequency $\dot{f}$ of a gravitational-wave signal, i.e., the spin-down, and can handle $|\dot{f}| \lesssim 10^{-12}$ Hz/s, even though our training sets only include signals with $\dot{f} = 0$. We evaluate the computational cost of our method to be $\mathcal{O}(10^{19})$ floating point operations, and compare it to those from standard all-sky searches, putting aside differences between covered parameter spaces. Our results show that our method is more efficient by 1 or 2 orders of magnitude than standard searches. Although our neural network takes about $\mathcal{O}(10^8)$ sec to employ using our current facilities [a single graphics processing unit (GPU) of GTX1080Ti], we expect that it can be reduced to an acceptable level by utilizing a larger number of improved GPUs.

DOI: 10.1103/PhysRevD.106.024025

I. INTRODUCTION

Gravitational waves that are characterized by quite long durations, quasimonochromatic frequencies, and almost constant amplitudes are called continuous gravitational waves (CGWs) (see [1–5] for reviews). In the source frame, the waveform model of a CGW is governed by a small number of parameters: an amplitude, an initial frequency, and a first (and higher) time derivative(s) of the frequency evolution.

Several sources are expected to emit CGWs, the most promising of which are distorted, rotating neutron stars. In radio astronomy, rotating neutron stars are already observed as pulsars [6], whose rotational frequencies and sky locations are accurately estimated according to the Australia Telescope National Facility Pulsar Database [7,8]. Therefore, this information can be used to look for CGWs emitted from these pulsars. This type of search is classified as a targeted search, because the source rotational frequency, its derivatives, and the sky position are known, which allows us to perform a deep, fully coherent analysis for CGWs. However, these searches are limited to $\mathcal{O}(100)$ known pulsars; thus, other types of analyses are needed for objects about which we have less information.

Supernova remnants could also be interesting sources of CGWs, since they could house a compact object, such as a neutron star, at its center. Although the rotation frequency of the central object is unknown, the source location can be (roughly) identified. Therefore, we can use directed search methods that do not assume a rotational frequency, which unfortunately requires a higher computational cost than that in targeted searches. Semicoherent methods had to be designed to make a search for these objects tractable, which

[*] yamamoto.takahiro.u6@f.mail.nagoya-u.ac.jp
[†] andrew.miller@uclouvain.be

reduces their sensitivity in comparison to that in targeted searches; however, the exciting possibility of discovering CGWs from something unknown motivates us to look at such systems.

The most difficult, computationally heavy search is called a blind search or all-sky search, in which we do not have any information about potential CGW sources, e.g., electromagnetically silent neutron stars, ultralight boson clouds around rotating black holes [9–13], or inspiraling binaries consisting of planetary-mass black holes [14–16] or of an ordinary compact object and a much lighter exotic one with an extreme mass ratio [17]. The computational difficulties in all-sky searches arise because we would observe CGWs in a moving detector frame. Thus, the observed phase of a CGW is modulated by Doppler effects due to the relative motion between the source and the detectors. And since we do not know where in the sky CGWs could come from, we must search each location individually, over all possible source parameters. Despite the computational cost, many clever methods (e.g., time-domain $\mathcal{F}$-statistic [18], frequency Hough [19], sky Hough [20], and hidden Markov model [21]) have performed all-sky searches resulting in competitive constraints on the amplitude of CGWs over the whole sky [22–30].

Another difficulty in CGW searches comes from the detector's non-Gaussian artifacts, which is known as line noise [31]. Line noise has an almost constant frequency and a much larger amplitude than the Gaussian component of the detector noise. Because of line noise, standard methods for all-sky searches are required to veto frequency bands where line noise is present. This makes the analysis blind around those frequencies since these lines are often spread into multiple frequency bins and have multiple harmonics. It is therefore necessary to devise algorithms that are not only computationally efficient but also robust against such artificial disturbances.

In the last five years, the application of deep learning has been widely discussed in gravitational-wave astronomy (see [32] for review), and there are several proposals on the application to the detection and parameter estimation of various sources [33–50]. Deep learning algorithms could provide a way of alleviating both high computational costs and extreme sensitivity to noise lines in all-sky searches. As we will show, neural networks could be trained on different noise artifacts, allowing a systemic discrimination between them and astrophysical signals. Furthermore, after training, neural networks can generally classify new data in different categories very quickly. Such methods could therefore be used alongside existing ones, allowing the standard searches to be performed with increased sensitivity.

In particular, for CGW searches, several groups have proposed deep learning to analyze long stretches of strain data with durations of $O(10^{5–7})$ sec, all of which treat the data differently before feeding them to the neural network. For example, Dreissigacker *et al.* [51] use the Fourier transformation to preprocess the data. They prepare several neural networks trained with the dataset corresponding to different frequency bands. Although the effects of non-Gaussian noise were not considered, they showed that their sensitivity is comparable to that of the semicoherent matched filter.

Combining deep learning with an existing analysis method is also a possible direction of research. Morawski *et al.* [52] employed the time-domain $\mathcal{F}$-statistic for each grid point in the parameter space as inputs to their neural network. Their network was constructed to classify the strain data into three classes: only Gaussian noise, CGW signal in Gaussian noise, and sinusoidal line with Gaussian noise. For data with a duration of two days, their method discriminated the aforementioned three cases with high accuracy but did not handle the case in which line noise and CGWs exist in the same data stream.

Beheshtipour and Papa [53] applied neural networks in the follow-up stages of the Einstein@Home pipeline in order to identify clusters of interesting candidates within the parameter space. They reported a slightly improved sensitivity compared to the results of an all-sky search in LIGO/Virgo's first observing run [23]; however, the computational cost was not reduced because Einstein@Home already requires a significant amount of computing power to perform the deepest all-sky searches in the CGW community.

Bayley *et al.* [54] combined deep learning and the Viterbi algorithm proposed in [21], and analyzed data with mock signal injections from the sixth science run of the initial LIGO/Virgo [55]. They found that their method achieves comparable sensitivity to semicoherent searches at a much lower computational cost. However, their neural networks specialized in detection and did not predict the source location.

Our previous work [56] demonstrates that a convolutional neural network can detect CGWs in a single detector output that contains stationary Gaussian noise. We proposed a new preprocessing method in which a double Fourier transform is applied to strain data that are partially demodulated by the time resampling. This preprocessing step concentrates the power of CGW signals into a small number of data points. The signal strength is therefore enhanced so that the neural network can easily detect CGWs. Our neural network independently treats the data of different source locations and frequency bins. Therefore, the computational cost of the follow-up can be reduced by specifying the parameter region where the follow-up is carried out. Although the sensitivity seems much better than that of the existing coherent search, it is demonstrated under the assumption that CGWs are circularly polarized (i.e., $\cos\iota = 1$ and $\psi = 0$). We no longer use this assumption in this work.

While there have been many efforts to use machine learning to detect CGWs, none of them systemically address the problem that non-Gaussian noise pollutes real

TABLE I. List of parameters characterizing the strain data and the preprocessing.

Description	Symbol	Value
Sampling frequency of the strain data	f_s	1024 (Hz)
Total duration of the strain data	T_dur	16777216 (sec)
Threshold of the phase modulation after the time resampling	$\delta\Phi_*$	0.01
The number of grid points	N_grid	5609178
Duration of SFT segment	T_seg	2048 (sec)
The number of data points within a SFT segment	L	2097152
Steepness parameter of Tukey window	ξ	0.125
The number of SFT segments	$N_\mathrm{seg}(=T_\mathrm{dur}/T_\mathrm{seg})$	8192
Upper limit of frequency which we analyze	f_up	100 (Hz)
The number of frequency bins	$N_\mathrm{bin}(=T_\mathrm{seg}f_\mathrm{up})$	204800

GW data containing astrophysical signals. Even those that have been applied to real data do not provide a recipe to handle non-Gaussian noise, nor do they indicate concretely how different line noise strengths affect their sensitivity and false alarm probability. The existing literature is therefore not systematic enough to be easily applied to future observing runs.

In this paper, we extend the work of [56] to the case in which the strain data are contaminated by line noise. Our current study demonstrates that our network can handle more realistic GW data, and that the preprocessing step enables the network to be robust against line noise. Furthermore, our method in which different frequencies are treated independently would be robust against “combs,” that is, a bunch of evenly spaced lines. We also show how the sensitivity degrades with increasing line noise strength, in comparison to that obtained in Gaussian noise, and how high spin-downs of a CGW could affect the sensitivity of our network. This paper demonstrates that neural networks must consider the impact of nonstationary noise in CGW searches, and provides a more realistic comparison of the performance of neural networks to other all-sky methods.

We organize the rest of the paper as follows: In Sec. II, we describe a waveform and line noise model, and how we process the strain data before feeding them into a convolutional neural network. In Sec. III, we explain our strategy to search for CGWs using a convolutional neural network. We show in Sec. IV sensitivity and false alarm probability estimations in the presence of Gaussian noise and Gaussian noise polluted by line noise, as well as robustness against small signal frequency changes. Following that, we estimate the computational cost of this method in Sec. V, and we make some concluding remarks and discuss ideas for future work in Sec. VI.

II. WAVEFORM MODELS AND PREPROCESSING

In this work, we assume that the spectral density of Gaussian noise is stationary. The total duration and the sampling frequency are denoted by T_dur and f_s, respectively. We fix them as $T_\mathrm{dur} = 2^{24}$ sec (∼192 days) and $f_\mathrm{s} = 1024$ Hz. See also Table I, which shows the parameters of the strain data and the preprocessing step.

A. Astrophysical signal

The observed signal depends on the antenna pattern and the phase evolution. The waveform that we observe can be written as [57]

$$h_\mathrm{obs}(t) \coloneqq h_0\left[F_+(t)\frac{1+\cos^2\iota}{2}\cos\Phi(t) + F_\times(t)\cos\iota\sin\Phi(t)\right], \tag{2.1}$$

where h_0 is the amplitude of the signal, ι is the inclination angle, and $F_+(t)$ and $F_\times(t)$ are the antenna pattern functions that depend on the source's location on the sky, the geometric configuration of the interferometer, and the location of the detector on Earth. The definitions of $F_+(t)$ and $F_\times(t)$ are the same as those used in Jaranowski *et al.* [18], and we assume a LIGO-Hanford detector [58]. $\Phi(t)$ is the observed phase of the gravitational waves, which we model to include the Doppler effect, the frequency f_gw, and the first time derivative of the frequency $\dot{f}$,

$$\Phi(t) = 2\pi f_\mathrm{gw}\left(t+\frac{\boldsymbol{r}(t)\cdot\boldsymbol{n}}{c}\right) + \pi\dot{f}\left(t+\frac{\boldsymbol{r}(t)\cdot\boldsymbol{n}}{c}\right)^2 + \phi_0, \tag{2.2}$$

where ϕ_0 is the initial phase, and $\boldsymbol{n}$ is the unit vector pointing to the source:

$$\boldsymbol{n}(\alpha,\delta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\epsilon & \sin\epsilon \\ 0 & -\sin\epsilon & \cos\epsilon \end{pmatrix}\begin{pmatrix} \cos\alpha\cos\delta \\ \sin\alpha\cos\delta \\ \sin\delta \end{pmatrix}. \tag{2.3}$$

Here, α is the right ascension, δ is the declination, and ϵ is the tilt angle between Earth's rotation axis and the orbital angular momentum. Here, we set the x axis to point toward the vernal equinox and the z axis to be along Earth's orbital angular momentum. We assume that the position vector of the detector $\boldsymbol{r}(t)$ can be decomposed into Earth's rotation, $\boldsymbol{r}_{\oplus}(t)$, and Earth's orbital motion $\boldsymbol{r}_{\odot}(t)$, i.e.,

$$\boldsymbol{r}(t) = \boldsymbol{r}_{\odot}(t) + \boldsymbol{r}_{\oplus}(t). \tag{2.4}$$

We neglect various effects, such as the orbital eccentricity and the influence of the other planets and the Moon, and assume that Earth follows a circular orbit on the xy plane. These effects would be taken into account properly by using a more sophisticated ephemeris in the preprocess stage. We write the orbital motion of Earth as

$$\boldsymbol{r}_{\odot}(t) = R_{\mathrm{ES}} \begin{pmatrix} \cos(\varphi_{\odot} + \Omega_{\odot} t) \\ \sin(\varphi_{\odot} + \Omega_{\odot} t) \\ 0 \end{pmatrix}, \tag{2.5}$$

where R_{ES}, $\Omega_{\odot}$, and $\varphi_{\odot}$ are the distance between Earth and the Sun, the angular velocity of the orbital motion, and the initial phase, respectively. The detector motion due to Earth's rotation is

$$\boldsymbol{r}_{\oplus}(t) = R_{\mathrm{E}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\epsilon & \sin\epsilon \\ 0 & -\sin\epsilon & \cos\epsilon \end{pmatrix} \begin{pmatrix} \cos\lambda \cos(\varphi_{\oplus} + \Omega_{\oplus} t) \\ \cos\lambda \sin(\varphi_{\oplus} + \Omega_{\oplus} t) \\ \sin\lambda \end{pmatrix}, \tag{2.6}$$

where R_{E}, λ, $\Omega_{\oplus}$, and $\varphi_{\oplus}$ are the radius of Earth, the latitude of the detector, the angular velocity of Earth's rotation, and the initial phase, respectively. In this work, we fix $\varphi_{\odot} = \varphi_{\oplus} = 0$ for simplicity.

B. Preprocessing

It is known that preprocessing the data plays a crucial role in improving a neural network's performance [59]. We employ the method proposed in the previous work [56], which we briefly review in this subsection.

Our method consists of three steps. In the first step, the strain data are transformed by a time resampling procedure. We prepare grid points on the sky, and define a new time coordinate for each grid point as

$$\tau(t; \alpha_a, \delta_a) \coloneqq t + \frac{\boldsymbol{r}(t) \cdot \boldsymbol{n}_a}{c}. \tag{2.7}$$

Here, a is the index specifying the grid point and runs from 1 to N_{grid}, the number of grid points on the sky. α_a and δ_a are the right ascension and the declination angle of the ath grid point, respectively, and $\boldsymbol{n}_a \coloneqq \boldsymbol{n}(\alpha_a, \delta_a)$ is the unit vector pointing to the ath grid point on the sky. Throughout the paper, the new time coordinate in Eq. (2.7) is abbreviated as τ_a. The resampled strain data are denoted by

$$s_a(\tau) \coloneqq s(t_a(\tau)), \tag{2.8}$$

where $s(t)$ is the strain data and $t_a(\tau)$ satisfies the relation

$$\tau = t_a(\tau) + \frac{\boldsymbol{r}(t_a(\tau)) \cdot \boldsymbol{n}_a}{c}. \tag{2.9}$$

The time grid is resampled such that in the new coordinate τ_a, the grid is uniformly spaced. Then, the phase modulation due to an astrophysical signal is removed, which results in signal power accumulating in a small number of frequency bins. Moreover, in the new time coordinate, sinusoidal line noise (see Sec. II C) is no longer monochromatic because it becomes Doppler modulated. That is why we expect an astrophysical signal can be discriminated from line noise.

The residual phase can be written as

$$\delta\Phi_a(t) = 2\pi f_{\mathrm{gw}} \frac{\boldsymbol{r}(t) \cdot \Delta\boldsymbol{n}_a}{c}, \tag{2.10}$$

where

$$\Delta\boldsymbol{n} \coloneqq \boldsymbol{n} - \boldsymbol{n}_a \tag{2.11}$$

is the deviation between the ath sky grid point and the gravitational-wave source. We split the residual phase into two parts: Earth's rotation

$$\delta\Phi_a^{\oplus}(t) \coloneqq 2\pi f_{\mathrm{gw}} \frac{\boldsymbol{r}_{\oplus}(t) \cdot \Delta\boldsymbol{n}_a}{c} \tag{2.12}$$

and Earth's orbital motion

$$\delta\Phi_a^{\odot}(t) \coloneqq 2\pi f_{\mathrm{gw}} \frac{\boldsymbol{r}_{\odot}(t) \cdot \Delta\boldsymbol{n}_a}{c}. \tag{2.13}$$

The grid points are placed on the sky such that the residual phase $\delta\Phi_a^{\oplus}(t)$ is suppressed below a threshold for any location of the source. This condition can be written as

$$\min_a \max_t |\delta\Phi_a^{\oplus}(t)| \leq \delta\Phi_* \text{ for any source}, \tag{2.14}$$

where $\delta\Phi_*$ is a threshold. As in the previous work, we use the template placement method proposed in Nakano *et al.* [60] to efficiently place grid points in a two-dimensional parameter space. First, we assume that the difference between the source direction and the grid point is small, and denote this difference by

$$\Delta\delta_a \coloneqq \delta - \delta_a, \qquad \Delta\alpha_a \coloneqq \alpha - \alpha_a. \tag{2.15}$$

We expand the residual phase $\delta\Phi_a^{\oplus}$ to the first order of $\Delta\alpha$ and $\Delta\delta$ as

$$\begin{aligned}\delta\Phi_a^{\oplus}(t) \simeq \frac{2\pi f_{\rm gw}}{c} R_{\rm E}\cos\lambda\{&-\Delta\delta_a \sin\delta_a \cos(\alpha_a - \varphi_{\oplus} - \Omega_{\oplus}t)\\ &- \Delta\alpha_a \cos\delta_a \sin(\alpha_a - \varphi_{\oplus} - \Omega_{\oplus}t)\},\end{aligned} \tag{2.16}$$

while neglecting the constant term. The maximum of $\delta\Phi_a^{\oplus}(t)$ is

$$\begin{aligned}\max_t|\delta\Phi_a^{\oplus}(t)| = &\frac{2\pi f_{\rm gw}}{c} R_{\rm E}\cos\lambda\\ &\times\sqrt{(\Delta\delta_a)^2\sin^2\delta_a + (\Delta\alpha_a)^2\cos^2\delta_a}.\end{aligned} \tag{2.17}$$

Allowing $\cos\lambda$ to be 1 makes the estimation of $\max_t|\delta\Phi_a^{\oplus}(t)|$ conservative. Therefore, we use $\cos\lambda = 1$ in the following. We rewrite the condition (2.14) as

$$\min_a[\Delta\sigma_a^2] \leq \delta\Phi_*^2\left(\frac{c}{2\pi f_{\rm gw}R_{\rm E}}\right)^2 \text{ for any source,} \tag{2.18}$$

with

$$\Delta\sigma^2 \coloneqq (\Delta\delta_a)^2\sin^2\delta_a + (\Delta\alpha_a)^2\cos^2\delta_a. \tag{2.19}$$

We define the metric on the two-dimensional parameter space (α, δ) as

$$\mathrm{d}\sigma^2 = e^{-2Y}(\mathrm{d}X^2 + \mathrm{d}Y^2), \tag{2.20}$$

with $X \coloneqq \alpha$ and $Y = -\log|\cos\delta|$. In this metric (2.20), the contour of $\Delta\sigma^2$ becomes a circle, which allows us to easily place grid points. In this work, we set the threshold at

$$\delta\Phi_* = 0.01. \tag{2.21}$$

The condition (2.18) depends on $f_{\rm gw}$. Here, we choose $f_{\rm gw} = 100$ Hz, which gives an upper bound of the frequency band $f_{\rm up}$. When we analyze this frequency band lower than $f_{\rm up}$, we do not need a new set of grid points because the residual phase cannot be larger than the threshold determined with $f_{\rm gw} = f_{\rm up}$. With these choices, we obtain

$$N_{\rm grid} = 5609178 \tag{2.22}$$

grid points to cover the entire sky.

The value of $\delta\Phi_*$ is arbitrarily chosen. If $\delta\Phi_*$ increases, the number of grid points is reduced, meaning that the computational cost for the preprocessing would decrease. But, signal power may be lost because a larger $\delta\Phi_*$ allows a larger residual phase. If $\delta\Phi_*$ is set to a lower value, the signal will be more visible, but the computational cost of the preprocessing would increase. We will return to this point in Sec. V.

In the second step, the short-time Fourier transformation (SFT) is applied to the resampled strains to make spectrograms. Because a spectrogram is generated for each grid point, the number of spectrograms is $N_{\rm grid}$. To avoid aliasing, each SFT segment is windowed by a Tukey window,

$$w[m] = \begin{cases}\frac{1}{2}\left[1-\cos\left(\frac{2\pi m}{\xi L}\right)\right], & 0 \leq m < \frac{\xi L}{2},\\ 1, & \frac{\xi L}{2} \leq m \leq L - \frac{\xi L}{2},\\ \frac{1}{2}\left[1-\cos\left(\frac{2\pi(L-m)}{\xi L}\right)\right], & L - \frac{\xi L}{2} < m \leq L.\end{cases} \tag{2.23}$$

Here, L is the window length and is given by

$$L \coloneqq T_{\rm seg}f_{\rm s}, \tag{2.24}$$

with the segment duration $T_{\rm seg}$. ξ is the parameter characterizing the steepness of the window edge, which we set to $\xi = 0.125$. A pixel value of a spectrogram is written as

$$\tilde{s}_{ak}[j] = \frac{1}{L}\sum_{m=0}^{L-1} w[m]s_a[jL+m]e^{-2\pi imk/L}. \tag{2.25}$$

Here, $s_a[m]$ is the discrete strain data defined by

$$s_a[m] \coloneqq s_a(m\Delta\tau), \tag{2.26}$$

with the time resolution $\Delta\tau = f_{\rm s}^{-1}$. We refer to the frequency corresponding to the kth frequency bin as f_k given by

$$f_k = k\Delta f, \tag{2.27}$$

where

$$\Delta f = \frac{1}{T_{\rm seg}} \tag{2.28}$$

is the frequency resolution of SFT. The number of the segment is denoted by $N_{\rm seg}$ and is given by

$$N_{\rm seg} = \frac{T_{\rm dur}}{T_{\rm seg}}. \tag{2.29}$$

An index j specifies a SFT segment. For a given grid point $\boldsymbol{n}_a$, a spectrogram (2.25) can be regarded as a set

$$\{\tilde{s}_{ak}|k = 1, 2, \ldots, N_{\rm bin}\} \tag{2.30}$$

of time series vectors

$$\tilde{s}_{ak} \coloneqq (\tilde{s}_{ak}[0], \tilde{s}_{ak}[1], \ldots, \tilde{s}_{ak}[N_{\rm seg} - 1]). \qquad (2.31)$$

Here, $N_{\rm bin}$ is the number of frequency bins we analyze and is given by

$$N_{\rm bin} = \frac{f_{\rm up}}{\Delta f}. \qquad (2.32)$$

Each vector $\tilde{s}_{ak}$ has an index corresponding to a frequency bin. If the time resampling procedure perfectly demodulates the gravitational-wave signal, the signal power is contained in one frequency bin that corresponds to the source frequency. Hence, we analyze the data in each frequency bin and each grid point $\boldsymbol{n}_a$ separately.

In the final step, another Fourier transform is applied to each time series vector. It is denoted by

$$\mathsf{S}_{ak}[\ell] \coloneqq \frac{1}{N_{\rm seg}} \sum_{j=0}^{N_{\rm seg}-1} \tilde{s}_{ak}[j] e^{-2\pi i j \ell / N_{\rm seg}}. \qquad (2.33)$$

In this expression, $\mathsf{S}_{ak}[\ell]$ for $0 \leq \ell \leq N_{\rm seg}/2 - 1$ expresses the positive frequency components, while the components of $N_{\rm seg}/2 \leq \ell \leq N_{\rm seg} - 1$ are filled by the negative frequency part. To align the vector component in ascending order of the frequency, we shift the components as

$$\begin{pmatrix} \mathsf{S}_{ak}[0] \\ \mathsf{S}_{ak}[1] \\ \vdots \\ \mathsf{S}_{ak}[N_{\rm seg}/2 - 1] \\ \mathsf{S}_{ak}[N_{\rm seg}/2] \\ \vdots \\ \mathsf{S}_{ak}[N_{\rm seg} - 1] \end{pmatrix} \to \begin{pmatrix} \mathsf{S}_{ak}[N_{\rm seg}/2] \\ \mathsf{S}_{ak}[N_{\rm seg}/2 + 1] \\ \vdots \\ \mathsf{S}_{ak}[N_{\rm seg} - 1] \\ \mathsf{S}_{ak}[0] \\ \vdots \\ \mathsf{S}_{ak}[N_{\rm seg}/2 - 1] \end{pmatrix}. \qquad (2.34)$$

For simplicity, we interpret the elements of a vector

$$\mathsf{S}_{ak} \coloneqq (\mathsf{S}_{ak}[0], \mathsf{S}_{ak}[1], \ldots, \mathsf{S}_{ak}[N_{\rm seg} - 1]), \qquad (2.35)$$

as already ordered by the transformation (2.34). Finally, we get a set

$$\{\mathsf{S}_{ak} | a = 1, 2, \ldots, N_{\rm grid}; k = 1, 2, \ldots, N_{\rm bin}\} \qquad (2.36)$$

from strain data. Figure 1 shows examples of preprocessed waveforms. The waveforms of Gaussian noise, astrophysical signals, and line noise are significantly different due to each waveform's response to the time resampling procedure. This procedure accumulates signal power in a small number of frequency bins; in contrast, it dilutes the power due to line noise in a wider frequency band.

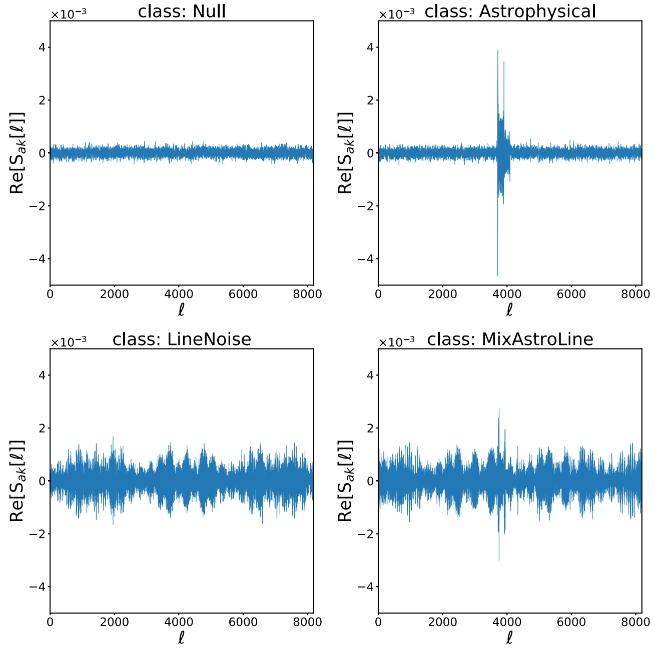

FIG. 1. Examples of the real part of the doubly Fourier transformed signals S_{ak}. From top left in the clockwise direction: only Gaussian noise, an astrophysical signal contaminated by Gaussian noise, an astrophysical signal contaminated by Gaussian noise and line noise, line noise and Gaussian noise. The amplitude of the astrophysical signal is set to $\log_{10} \hat{h}_0 = -1.0$ and the line noise is $\log_{10} \hat{h}_0^{\rm line} = 0.0$ [the amplitude parameters $\hat{h}_0$ and $\hat{h}_0^{\rm line}$ are defined in Eqs. (3.8) and (3.9), respectively]. Note that these values are an optimistic case. After the second Fourier transform, we shift ℓ as shown in Eq. (2.34). Therefore, if the GW frequency is close to the frequency of the SFT bin, the GW signal forms an excess around the center.

Now, we discuss the noise properties after the time resampling. We assume that the detector noise is stationary and Gaussian, and the time resampling does not change the statistical properties of the detector noise. The noise power spectrum $S_{\rm n}(f)$ of the detector noise is defined by

$$\langle \tilde{n}(f) \tilde{n}^*(f') \rangle = \frac{1}{2} S_{\rm n}(f) \delta(f - f'), \qquad (2.37)$$

and we have defined the Fourier transform as

$$\tilde{n}(f) = \int_{-\infty}^{\infty} {\rm d}f n(t) e^{-2\pi i f t}. \qquad (2.38)$$

After whitening, the noises in different SFT segments are independent. Therefore, the noise power spectrum density is

$$\langle \tilde{n}_{ak}[j] \tilde{n}_{ak}^*[j'] \rangle = \frac{1}{2T_{\rm seg}} \delta_{jj'} \qquad (2.39)$$

for any grid point a. Here, we denote the detector's Gaussian noise after the time resampling by

$$n_a(\tau) \coloneqq n(t_a(\tau)), \tag{2.40}$$

and its Fourier transform by

$$\tilde{n}_{ak}[j] = \frac{1}{L}\sum_{m=0}^{L-1} n_a[jL+n]e^{-2\pi imk/L}. \tag{2.41}$$

We neglect the effect of the Tukey window. The Fourier transform in Eq. (2.41) differs from Eq. (2.38) by the normalization factor of $1/L$. Similar to Eq. (2.33), we define

$$\mathsf{N}_{ak}[\ell] \coloneqq \frac{1}{N_{\mathrm{seg}}}\sum_{j=0}^{N_{\mathrm{seg}}-1} \tilde{n}_{ak}[j]e^{-2\pi ij\ell/N_{\mathrm{seg}}}. \tag{2.42}$$

The variance of $\mathsf{N}_{ak}[\ell]$ can be obtained as

$$\begin{aligned}\langle \mathsf{N}_{ak}[\ell]\mathsf{N}^*_{ak}[\ell']\rangle &= \frac{1}{N_{\mathrm{seg}}^2}\sum_{j,j'}\langle \tilde{n}_{ak}[j]\tilde{n}^*_{ak}[j']\rangle e^{-2\pi i(j\ell-j'\ell')/N_{\mathrm{seg}}}\\ &= \frac{1}{2N_{\mathrm{seg}}T_{\mathrm{seg}}}\delta_{\ell\ell'}.\end{aligned} \tag{2.43}$$

We generate simulated Gaussian noise in the transformed strain data by using Eq. (2.43) [61].

C. Line noise

Line noises are usually classified into three types: (1) perfectly sinusoidal line noise, (2) sinusoidal line noise with finite coherence time, and (3) comb line noise. Perfectly sinusoidal line noise is modeled by a sinusoidal function with a constant frequency. This is the simplest model of a line noise.

Perfectly sinusoidal line noise is modeled by

$$n^{\mathrm{sin}}_{\mathrm{line}}(t) = n_0\cos(2\pi f_{\mathrm{line}}t + \phi_0), \tag{2.44}$$

where n_0 is the line amplitude, f_{line} is the line noise frequency, and ϕ_0 is the initial phase. The model (2.44) does not have a frequency modulation or an amplitude modulation. The spectral density has an infinitely narrow peak at the frequency f_{line}.

In practice, the frequency of line noise can change on a certain timescale. If line noise has a finite coherence time, the power of the line noise dissipates in a wide range of frequency bins. It leads to the suppression of line noise power contained in a preprocessed vector S_{ak}. In this work though, for simplicity, we do not account for line noise with a finite coherence time.

Some instrumental disturbances could also cause multiple line noises with different frequencies that are evenly spaced, i.e., combs. However, for most combs observed in the first and second observing runs of Advanced LIGO and Advanced Virgo [31], the spacing in frequency is much larger than the Doppler modulation. Therefore, each of the comb's teeth would be contained in a different frequency bin and would safely be regarded as a single line, so we can focus here on perfectly sinusoidal line noise.

Let us remark on the amplitude of the line noise we consider in the presented work. As we stated above, lines are assumed to be stable and monochromatic. They can be removed by whitening if their amplitudes are much larger than the Gaussian noise level. In reality though, more sophisticated methods are required to remove lines because they have finite coherent times. Even stable lines cannot be completely removed if their amplitudes are comparable to the Gaussian noise level. Therefore, in this work, we assume the line noise amplitude to be in the range

$$1.0 \leq n_0\left(\frac{S_{\mathrm{n}}(f_k)}{1\ \mathrm{Hz}^{-1}}\right)^{-1/2} \leq 10.0. \tag{2.45}$$

III. METHOD

We use deep learning to discriminate the presence or absence of a GW signal and/or a sinusoidal line in Gaussian noise. The fundamentals of deep learning are summarized in Appendix A. In deep learning, we can use a neural network to extract data features and give a prediction for newly obtained data. Here, we construct a convolutional neural network (CNN) to classify the input vectors S_{ak} into four classes: (1) only Gaussian noise (`Null`), (2) astrophysical signal injected into Gaussian noise (`Astrophysical`), (3) sinusoidal line noise injected into Gaussian noise (`LineNoise`), and (4) astrophysical signal in the presence of both sinusoidal line noise and Gaussian noise (`MixAstroLine`). Our CNN is trained to predict the probabilities (A8) that certain strain data fall into each class.

Using the probabilities that the CNN has predicted, we then need to decide on a definition of "detection" of an astrophysical signal. Here, we choose the standard criterion; the data are classified into the class for which the CNN gives the largest probability. We assume that a vector contains an astrophysical signal if the CNN classifies the vector as `Astrophysical` or `MixAstroLine`, while we try another definition of detection later.

We use the term "candidates" to indicate a set of vectors determined to have an astrophysical signal based on the procedure described above. Each candidate is characterized by a SFT frequency bin and a grid point, as well as the vector S_{ak}, whose indices describe the frequency bin k and the grid point a.

A. CNN architecture

Table II shows the structure of the CNN we used. It consists of six convolutional layers, three max-pooling layers, and three fully connected layers. In the table, a rectified linear unit (ReLU) transformation is counted as a

TABLE II. Structure of the CNN we used in this work. The first column shows the types of layers. Here, we separately list the activation functions. Roughly speaking, the CNN can be divided into two blocks. The first block consists of convolutional layers, pooling layers, and activation functions. The second block comprises the fully connected layers, activation functions, and a softmax layer. Before the first fully connected layers, the transformation called flattening is applied. It transforms a two-dimensional tensor into a one-dimensional vector. The second column shows the output size of the layer. For the layers before the flattening, the output is a two-dimensional tensor and its shape is described by two numbers. The first number shows the number of channels, while the second number is the length of data. The third column gives the kernel sizes of the convolutional layers and the pooling layers, while the last layer shows the number of tunable parameters. The first row shows the input vector that has the length of 8192 and two channels. The CNN has six convolutional layers. The number of tunable parameters is calculated by Eqs. (A3) and (A6). The total number of tunable parameters is 4171170.

Layer	Output size	Kernel size	No. of parameters
(Input)	(2, 8192)	–	–
1D convolutional	(16, 8177)	16	528
ReLU	(16, 8177)	–	
1D convolutional	(16, 8162)	16	4112
ReLU	(16, 8162)	–	–
Max pooling	(16, 2040)	4	–
1D convolutional	(32, 2033)	16	4128
ReLU	(32, 2033)	–	–
1D convolutional	(32, 2026)	16	8224
ReLU	(32, 2026)	–	–
Max pooling	(32, 506)	4	–
1D convolutional	(64, 503)	4	8256
ReLU	(64, 503)	–	–
1D convolutional	(64, 500)	4	16448
ReLU	(64, 500)	–	–
Max pooling	(64, 125)	4	–
Flattening	(8000,)	–	–
Fully connected	(512,)	–	4096512
ReLU	(512,)	–	–
Fully connected	(64,)	–	32832
ReLU	(64,)	–	–
Fully connected	(4,)	–	260
Softmax	(4,)	–	–

layer, and a linear transform in the fully connected layer is separated from the activation.

The CNN takes the real part and the imaginary part of a vector S_{ak} as an input. Respecting Eq. (A5), we write the input vector as

$$x_{1j} = \mathrm{Re}[\mathsf{S}_{ak}[j]], \qquad x_{2j} = \mathrm{Im}[\mathsf{S}_{ak}[j]]. \tag{3.1}$$

It is known that normalizing input data accelerates and stabilizes the training [59]. We normalize the input vector so that it has a mean of 0 and a standard deviation of unity:

$$\hat{x}_{aj} = \frac{x_{aj} - \mu}{\sigma}, \tag{3.2}$$

where the mean is given by

$$\mu \coloneqq \frac{1}{2N_{\mathrm{in}}} \sum_{a=1,2} \sum_{j=1}^{N_{\mathrm{in}}} x_{aj}, \tag{3.3}$$

and the standard deviation is

$$\sigma \coloneqq \sqrt{\frac{1}{2N_{\mathrm{in}}} \sum_{a=1,2} \sum_{j=1}^{N_{\mathrm{in}}} (x_{aj} - \mu)^2}. \tag{3.4}$$

We employ the cross-entropy loss function (A10), use the Adam optimzer [62], and implement the CNNs within the deep learning library PYTORCH [63]. The training and evaluation are carried out with a single graphics processing unit (GPU) GeForce GTX1080Ti. We trained the CNN for 300 epochs, and do not observe overfitting. Therefore, we use the CNN state at the end of the training.

B. Data preparation

In our work, we generate datasets in a limited frequency band, and the results (e.g., sensitivity, false alarm rate) are extrapolated lower frequencies. We use the frequency band of

$$f_k - \frac{1}{2}\Delta f_{\mathrm{gw}} \leq f_{\mathrm{gw}} \leq f_k + \frac{1}{2}\Delta f \tag{3.5}$$

with

$$f_k = 100\ \mathrm{Hz}. \tag{3.6}$$

Equation (3.5) is the width of the kth frequency bin corresponding to 100 Hz.

In this work, we focus only on the selected frequency bin (i.e., $f_k = 100$ Hz) and train on simulated monochromatic signals,

$$\dot{f} = 0. \tag{3.7}$$

The sensitivity of the method is quantified by determining the minimum amplitude that an injected signal could be detected with a given detection probability and a specified false alarm probability. If we simply use h_0 as an indicator of the method's sensitivity, the sensitivity is affected by the detector noise level. To avoid such an effect, we normalize the amplitude as

$$\hat{h}_0 \coloneqq h_0 \left(\frac{S_{\mathrm{n}}(f_k)}{1\ \mathrm{Hz}^{-1}} \right)^{-1/2}, \tag{3.8}$$

TABLE III. Source parameters. The range of normalized amplitudes is chosen so that it covers (1) the minimum limit of the detectable signal, and (2) sufficiently large signals so that the CNN can learn the signals efficiently.

Description	Distribution
Normalized amplitude $\hat{h}_0$	Log uniform on $[10^{-2}, 10^1]$
Frequency $f_{\rm gw}$	Uniform on $[f_k - \frac{\Delta f}{2}, f_k + \frac{\Delta f}{2}]$
Inclination angle ι	Uniform on $[0, \pi]$
Right ascension α	Uniformly distributed on the sky
Declination angle δ	Uniformly distributed on the sky
Polarization angle ψ	Uniform on $[0, 2\pi]$
Initial phase ϕ_0	Uniform on $[0, 2\pi]$

where $S_{\rm n}(f_k)$ is the power spectral density of the detector's Gaussian noise at the reference frequency f_k. We use the normalized amplitude (3.8) when we generate the dataset and quantify our CNN's sensitivity.

Under this assumption, the gravitational-wave signal can be characterized by seven parameters, i.e., a normalized amplitude $\hat{h}_0$, a frequency $f_{\rm gw}$, an inclination angle ι, a right ascension α, a declination angle δ, a polarization angle ψ, and an initial phase ϕ_0. Table III shows the distributions of source parameters that we sampled from to generate the training and the validation datasets. For the normalized amplitude, the upper limit of the range is set to be slightly larger than what we expected in realistic situations. The neural network can learn the features of an astrophysical signal from data that contain signals with large amplitudes and gradually become able to capture the signature of lower amplitude signals. As for the source locations, we uniformly distribute the position on the sky. Here, we assume that the GW signal is significantly suppressed if the Doppler correction is not appropriate. Such a situation occurs when our chosen grid points lie far away from the source location. Therefore, when we generate the data of the Astrophysical and MixAstroLine classes, we pick the closest grid point to the source location.

We use Eq. (2.44) as the line noise model. It is characterized by an amplitude n_0, a frequency $f_{\rm line}$, and an initial phase ϕ_0. The frequency $f_{\rm line}$ is also limited to the range of Eq. (3.5). The normalized amplitude (3.8) is employed instead of n_0, i.e.,

$$\hat{h}_0^{\rm line} \coloneqq n_0 \left(\frac{S_{\rm n}(f_k)}{1\ {\rm Hz}^{-1}} \right)^{-1/2}. \tag{3.9}$$

Table IV shows the parameters characterizing line noise and how they are sampled. After generating sinusoidal line noise, we transform it using the time resampling procedure with the randomly chosen grid points.

We prepare 20000 GW signals, 20000 sinusoidal lines, and 20000 pairs of GW signals and lines for training. They correspond to the Astrophysical, LineNoise, and MixAstroLine classes, respectively. In generating GW signals, we use the fact that the extrinsic parameters (amplitude, polarization angle, inclination angle, and initial phase) can be factored out of the CGW waveform. We therefore generate the waveform that depends only on $f_{\rm gw}$ and source position. For each iteration of training, we sample extrinsic parameters and include the effects of extrinsic parameters to determine the CGW waveform.

TABLE IV. Line noise parameters.

Description	Distribution
Normalized amplitude $\hat{h}_0^{\rm line}$	Log uniform on [1, 10]
Frequency $f_{\rm line}$	Uniform on $[f_k - \frac{\Delta f}{2}, f_k + \frac{\Delta f}{2}]$
Initial phase ϕ_0	Uniform on $[0, 2\pi]$

A similar factorization can be done for the line noise waveform. We can factor out the amplitude and multiply the waveform by a randomly selected one in each iteration. This means that the line noise waveform depends only on $f_{\rm line}$. Before feeding the waveforms into the CNN, we inject them into Gaussian noise with the variance given by Eq. (2.43).

For the Null class, we only give Gaussian noise to the CNN. Thus, we do not need to generate any data for the Null class in advance. The validation data are generated by the same procedure, but the number of data is decreased to 2000 for each class.

IV. RESULTS

Figure 2 shows the confusion matrix of the trained CNN. We use 2000 test data for each class. The amplitude of gravitational wave is uniformly sampled from $\log_{10} \hat{h}_0 \in [-2.0, -1.0]$, and that of line noise is sampled from $\log_{10} \hat{h}_0^{\rm line} \in [0.0, 1.0]$. Here, we changed the range of the signal amplitude from that employed in the training data because we want to test our CNN for data with realistic

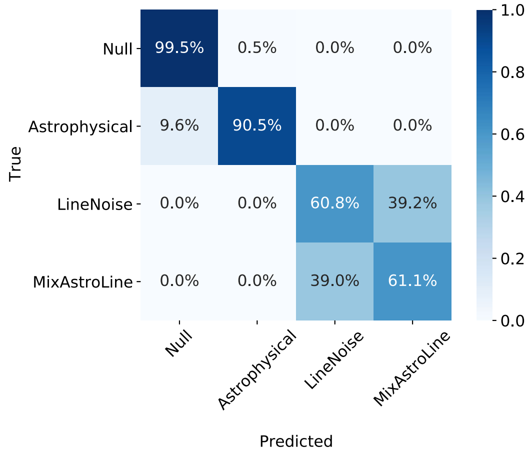

FIG. 2. Confusion matrix of the CNN. This matrix quantifies the fraction of testing data that were classified correctly (diagonal elements) and incorrectly (off-diagonal elements).

TABLE V. Result for the case where only Gaussian noise exists. We use 20000 test events that contain only simulated Gaussian noise. Our CNN can classify the Gaussian noise data with accuracy of 99.34%. The false alarm probability is 0.66%.

Predicted class	No. of events	Fraction (%)
Null	19868	99.34
Astrophysical	132	0.66
LineNoise	0	0.0
MixAstroLine	0	0.0

amplitudes. Most of the data in the `Null` class are correctly classified as the `Null class`. The significant point of Fig. 2 is that the CNN can discriminate between the presence and the absence of line noises. The test events of the `Null` class and the `Astrophysical` class are not misclassified as the `LineNoise` class or the `MixAstroLine` class. On the contrary, the test data containing line noise are classified in the `LineNoise` class or the `MixAstroLine` class, and there is some confusion with the `Null` class. From these results, it can be concluded that the CNN can tell apart a line from its absence.

From the top row of Fig. 2, it is found that only 0.5% of events that contain just Gaussian noise are classified as the `Astrophysical` class. Furthermore, we use 20000 simulated Gaussian noise data to evaluate the false alarm probability for the Gaussian noise, as shown in Table V. Among 20000 test noise data, 132 events are classified as the `Astrophysical` class. The estimated false alarm probability to misclassify Gaussian noise as an astrophysical signal is 0.66%, which is comparable to that estimated by 2000 test events. Even for 20000 events, we find no confusion between the line noise class and the mixed class.

We study the detailed result for the `Astrophysical` class. With varying the normalized amplitude $\log_{10}\hat{h}_0$ from -2.0 to -1.0 with the step of 0.1, we prepare 11 datasets corresponding to the respective values of the amplitude. Each dataset consists of 2000 injections. We apply the trained CNN to each dataset and count the number of detected events for each predicted class. Figure 3 shows the fraction of events as a function of the normalized amplitude. The detection probability exceeds 95% for $\log_{10}\hat{h}_0 \gtrsim -1.64$. We also quote our results in terms of the so-called sensitivity depth, which is defined as

$$\mathcal{D} \coloneqq \frac{\sqrt{S_\mathrm{n}(f_k)/1\ \mathrm{Hz}^{-1}}}{h_0} = (\hat{h}_0)^{-1}. \tag{4.1}$$

In terms of the sensitivity depth, our CNN has a sensitivity of

$$\mathcal{D}^{95\%} \simeq 43.9. \tag{4.2}$$

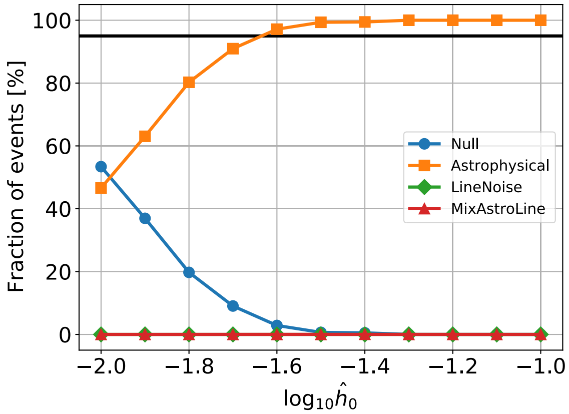

FIG. 3. Detection efficiency of the CNN for astrophysical signals injected into Gaussian noise. The horizontal axis shows the logarithm of the amplitude, and the vertical axis is the fraction of events. Orange squares indicate the detection probability of astrophysical signals. For $\log_{10}\hat{h}_0 \gtrsim -1.64$, the detection probability exceeds 95%. The detection probability decreases as the amplitude decreases. The results of the `LineNoise` class (green diamonds) and the `MixAstroLine` class (red triangles) are overlapped.

The LIGO/Virgo Collaboration has carried out all-sky searches for isolated neutron stars using data from LIGO/Virgo's third observation data [30], which results in upper limits on the gravitational-wave strain amplitude. We compare the sensitivity depths of the standard methods and our method in Table VI. It shows that our neural network can outperform the time-domain $\mathcal{F}$-statistic and the Viterbi algorithm assisted by deep learning (SOAP). Furthermore, our method has comparable sensitivity to the frequency Hough and the sky Hough. We emphasize, however, that they search over different parameter spaces: The standard method surveys a wide range of $\dot{f}$, while our method focuses on quasimonochromatic waves.

TABLE VI. Comparison of the sensitivity depths of the standard all-sky search methods and our method. For frequency Hough and time-domain $\mathcal{F}$-statistic, the upper limits on the amplitude $h_0^{95\%}$ are presented in [30]. We converted them into $\mathcal{D}^{95\%}$ assuming $\sqrt{S_\mathrm{n}(f)} = 5.2 \times 10^{-24}$ [Hz^{-1}] that is shown in Fig. 6 of [30]. For sky Hough and time-domain $\mathcal{F}$-statistic, we read the values, respectively, from Figs. 11 and 13 of [30] that show their upper limit on the amplitude. We stress that the parameter region and the strain duration are different depending on the method.

Method	Frequency band	$\mathcal{D}^{95\%}$
Frequency Hough	At 100 Hz	42–43
Sky Hough	At 116.5 Hz	47.2
Time-domain $\mathcal{F}$-statistic	At 100 Hz	26–52
SOAP	On 40–500 Hz	9.9
Our method	$\lesssim$100 Hz	43.9

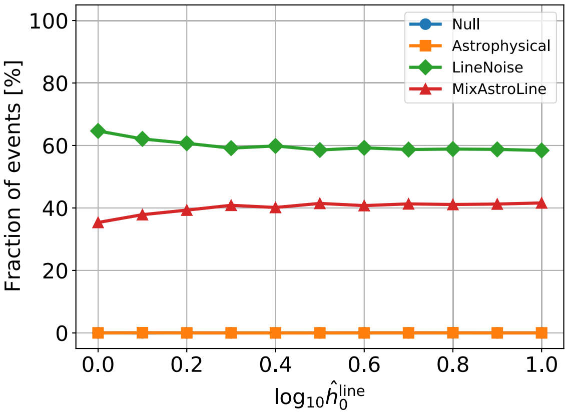

FIG. 4. Classification results for test data containing only line noise with Gaussian noise. For any amplitude, the fraction of correctly classified events is about 60% (green diamonds). The misclassification as the `MixAstroLine` class (red triangles) occurs for 40% of test data. The number of misclassifications as the `Null` class (blue circles) and the `Astrophysical` class (orange squares) is almost zero. Their markers are overlapped.

The duration of the signal is also different; O3 data have the duration of ∼11 months $\sim 2.9\times10^7$ sec, and our method assumes that signals last for $2^{24}\sim1.6\times10^7$ sec.

Whereas the `Astrophysical` class and the `Null` class are classified correctly, the events contaminated by the line noise are not. The false alarm probability that the line noise data are classified in the `MixAstroLine` class is estimated to be 39.2%. To test the CNN for line noise data, we prepare 11 datasets corresponding to different amplitudes of the line noise. Each dataset contains 2000 lines injected into Gaussian noise. Figure 4 shows the classification result for the test data of the `LineNoise` class as a function of the line noise amplitude $\log_{10}\hat{h}_0^{\rm line}$. The classification results are almost constant for any value of $\log_{10}\hat{h}_0^{\rm line}$. We can interpret this result as follows: We have injected line noise events with amplitudes much larger than the Gaussian noise. Therefore, the overall amplitude of the line noise would disappear by normalization [see Eq. (3.2)], with the result that the sensitivity of the CNN does not depend on the line noise amplitude, as shown in Fig. 4.

While the CNN can discriminate the presence and the absence of a line, it cannot find the astrophysical signal when line noise contaminates. As shown in Fig. 2, line noise is misclassified as the `MixAstroLine` with the false alarm probability of ∼40%. The false alarm probability could be suppressed by changing the detection criterion. As stated in the Sec. III, we define a detection as when the predicted probability of the `MixAstroLine` (or `Astrophysical`) class dominates others. Here, we introduce a new criterion given by

$$p_{\rm th}\le p_{\rm Mix}, \tag{4.3}$$

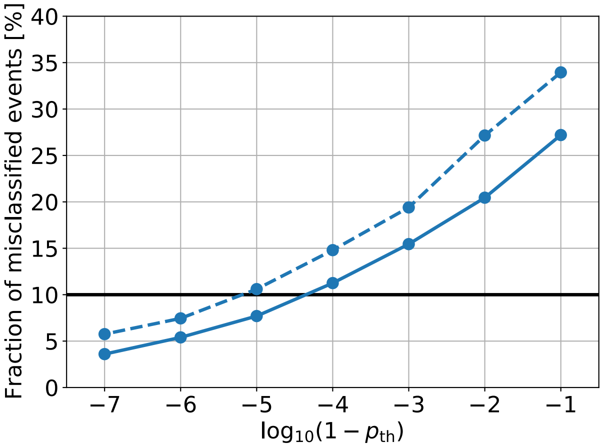

FIG. 5. False alarm probabilities as a function of the threshold $p_{\rm th}$. The horizontal axis corresponds to $p_{\rm th}$. The vertical axis shows the fraction of `LineNoise` events which are misclassified as `MixAstroLine`. The dashed lines and the solid lines, respectively, present the cases of $\log_{10}\hat{h}_0^{\rm line}=0.0$ and 1.0. Black horizontal line corresponds to the misclassification probability of 10%. If we set $p_{\rm th}=1\text{–}10^{-6}$, we can suppress the false alarm probability less than 10%.

where $p_{\rm Mix}$ is the CNN predicted probability of the `MixAstroLine` class. Figure 5 shows the false alarm probabilities with various values of $p_{\rm th}$. In order to achieve a false alarm probability that is less than 10% for data contaminated by line noise, we need to set $p_{\rm th}=1\text{–}10^{-6}$. This detection threshold is used in the rest of the paper.

Figure 6 shows the detection efficiency of the `MixAstroLine` signals. Comparing to the case where the line noise is absent, the efficiency is degraded because of the line noise. We estimate the sensitivity depth

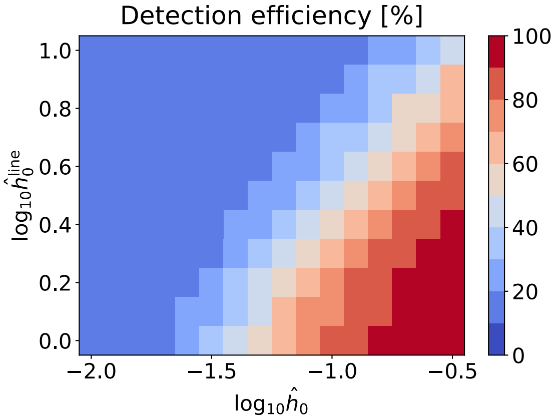

FIG. 6. Detection probabilities of an astrophysical signal coexisting with line noise. The horizontal and vertical axes show the normalized amplitudes of astrophysical signals and line noise, respectively. In this figure, we set the threshold $p_{\rm th}=1\text{–}10^{-6}$. In most regions, the detection probabilities are less than 50%. The maximum detection probability is 96.1% at $(\log_{10}\hat{h}_0,\log_{10}\hat{h}_0^{\rm line})=(-0.5,0.0)$.

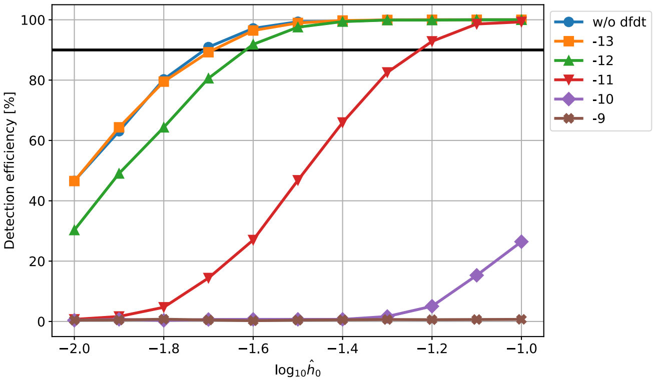

FIG. 7. Detection probability of the signals with nonzero frequency derivatives. For $\dot{f} = -1.0\times10^{-13}$ Hz/ sec, which is shown by orange squares, the sensitivity is not degraded compared with the $\dot{f}=0$ case (blue circles). The detection probability starts to diminish from $\dot{f} = -1.0\times10^{-12}$ Hz/ sec (green up triangles). For $|\dot{f}| \lesssim 10^{-11}$ Hz/ sec, the sensitivity is significantly reduced.

$$\mathcal{D}^{95\%} \simeq 3.62 \quad \text{for } \log_{10}\hat{h}_0^{\text{line}} = 0.0, \tag{4.4}$$

which is only ∼8.2% of that in the absence of the line noise [see Eq. (4.2)].

Realistic gravitational-wave sources naturally have intrinsic frequency evolution as they are modeled in Eq. (2.2). Therefore, we test the four-class CNN also for signals with nonzero $\dot{f}$. Different datasets are generated, each with the fixed $\dot{f}$: $\dot{f} = -10^{-13}$, -10^{-12}, -10^{-11}, -10^{-10}, and -10^{-9} Hz/ sec. For each $\dot{f}$, we prepare 2000 test data and evaluate the detection probability, and show the classification results in Fig. 7. For the data with $|\dot{f}|$ smaller than 10^{-12} Hz/ sec, the CNN's performance is not much degraded. Especially for $\dot{f} = -10^{-13}$ Hz/ sec, the detection probability is comparable to that of the $\dot{f}=0$ case for all amplitudes. On the other hand, the performance becomes worse as the frequency derivative exceeds $|\dot{f}| = 10^{-11}$ Hz/ sec. It can be understood as follows: As explained in Sec. II, the input data should contain the signal power with a SFT frequency bin. With nonzero $\dot{f}$, however, the frequency track might cross a number of frequency bins, spreading the signal power over multiple frequency bins. We therefore expect that signals with higher $\dot{f}$ cannot be detected as efficiently by the CNN as those with lower $\dot{f}$.

Quantitatively, the frequency width of a bin is

$$\Delta f = \frac{1}{T_{\text{seg}}} \simeq 4.88\times10^{-4}\ \text{Hz}. \tag{4.5}$$

The frequency change from the initial time across T_{dur} can be estimated as

$$\delta f \sim T_{\text{dur}}\dot{f} \sim 10^{-4}\ \text{Hz}\left(\frac{\dot{f}}{10^{-11}\ \text{Hz/ sec}}\right). \tag{4.6}$$

Roughly speaking, if $\delta f \lesssim \Delta f$, the signal power is still contained in one frequency bin. Thus, we expect that the CNN is applicable with comparable accuracy to that achieved in the $\dot{f}=0$ case. On the other hand, if $\Delta f \lesssim \delta f$, the signal power dissipates into several frequency bins. Thus, the CNN's performance degrades when $|\dot{f}| \gtrsim 10^{-11}$ Hz/ sec.

V. COMPUTATIONAL COST

In this section, we evaluate the computational cost of each processing step. First, we estimate the computational cost of the preprocess. The most expensive part of the preprocess is the SFT for making the spectrogram and the Fourier transform to obtain a set of vectors $\{\mathsf{S}_{ak}\}$. We assume that the cost of the time resampling is negligible compared to the SFT and the Fourier transform. For each grid point, we perform the SFT and the Fourier transform. The computational cost of taking SFTs can be estimated by

$$\mathcal{N}_{\text{SFT}} = N_{\text{seg}}\cdot 5f_{\text{s}}T_{\text{seg}}\log_2[f_{\text{s}}T_{\text{seg}}]. \tag{5.1}$$

Here, we evaluate the number of data points contained in a SFT segment as $f_{\text{s}}T_{\text{seg}}$. Using the values listed in Table I, we estimate

$$\mathcal{N}_{\text{SFT}} \simeq 1.80\times10^{12} \tag{5.2}$$

in the unit of the number of floating point operations. Similarly, the computational cost of the Fourier transform for achieving a set of vectors $\{\mathsf{S}_{ak}\}$ can be estimated as

$$\begin{aligned}\mathcal{N}_{\text{Fourier}} &= N_{\text{bin}}(5N_{\text{seg}}\log_2 N_{\text{seg}}) \\ &\simeq 1.09\times10^{11}.\end{aligned} \tag{5.3}$$

Combining Eqs. (5.1) and (5.3), we obtain the computational cost of the preprocess,

$$\begin{aligned}\mathcal{N}_{\text{preprocess}} &= N_{\text{grid}}(\mathcal{N}_{\text{SFT}} + \mathcal{N}_{\text{Fourier}}) \\ &\simeq 1.07\times10^{19}.\end{aligned} \tag{5.4}$$

We evaluate the computational time of the CNN by extrapolating the measured value for a small subset consisting of the test data. With a single GPU (GTX1080Ti), we measure the computational time to process 10^5 data five times. We obtained their averaged time of 8.8742 sec and the standard deviation of 0.0237 sec. The total number of the vectors $\{\mathsf{S}_{ak}\}$ to be processed is

$$N_{\text{vec}} = N_{\text{grid}}\cdot N_{\text{bin}} = 1.15\times10^{12}. \tag{5.5}$$

TABLE VII. Comparison of the computational time of the standard methods and our method. We estimate the core hour with the spec of Intel E5-2670; the clock frequency is 2.6 GHz, eight operations per clock leading to the computational speed of 20.8 GFlops per core. This computational time includes only floating-point operations. We take these values from [25], except that the computational time of SOAP is taken from [54]. We do not consider input/output (I/O) time.

Method	Corehour
Frequency Hough	9×10^6
Sky Hough	2.5×10^6
Time-domain $\mathcal{F}$-statistic	2.4×10^7
SOAP	$1\text{–}2 \times 10^2$
Our method	1.4×10^5

Therefore, the estimated time to process all vectors is

$$T_{\mathrm{CNN}} \simeq 1.02 \times 10^8 \ [\mathrm{sec}]. \tag{5.6}$$

Although this is longer than the total duration T_{dur} by an order of magnitude, we expect this can be suppressed to a negligible level by taking into account the development of hardware and the use of multiple GPUs in parallel [64].

In Table VII, we compare the computational cost, in units of core hours, of our method to that from the standard all-sky search pipelines employed in LIGO/Virgo's second observing run [25]. As in [25], we assume the hardware Intel E5-2670 that has a clock frequency of 2.6 GHz and carries out eight floating-point operations per clock. The computational speed is 20.8 GFlops per core. Using Eq. (5.4), we estimate the computational time by

$$\frac{\mathcal{N}_{\mathrm{preprocess}}}{20.8\ [\mathrm{GFlops}]} \simeq 1.4 \times 10^5 \ [\mathrm{corehr}]. \tag{5.7}$$

It shows that our method is computationally more efficient by 1 or 2 orders of magnitude than the standard methods in which deep learning is not employed. Again, we stress that the parameter region and the duration of the strain data are different depending on the method.

Before ending this section, we mention how the computational cost of the preprocess depends on the various parameters governing our method. We focus on three parameters: the phase resolution $\delta\Phi_*$, the duration of the SFT segment T_{seg}, and the upper bound of the frequency band we explore, f_{up}. Figure 8 shows the computational cost of the preprocess with various values of $\delta\Phi_*$, T_{seg}, and f_{up}. To create this figure, we assume that the sampling frequency is set to $f_s = 10 f_{\mathrm{up}}$. From this figure, we need to choose a low f_{up}, a short T_{seg}, and a high $\delta\Phi_*$ in order to reduce the computational cost of the preprocess. In the standard methods, f_{up} is usually set to $\sim 10^3$ Hz. We can make our method applicable for such frequency bands by choosing T_{seg} and $\delta\Phi_*$ appropriately. If we want to suppress the computational time to 10^7 core hour to explore the frequency band up to 10^3 Hz, we should set $T_{\mathrm{seg}} \simeq 100$ sec or 400 sec for $\delta\Phi_* = 0.01$ and 0.02, respectively. Changing the parameters affects not only the computational cost but also the performance of our CNN. It is important to study the dependence of the performance, but we leave it as future work.

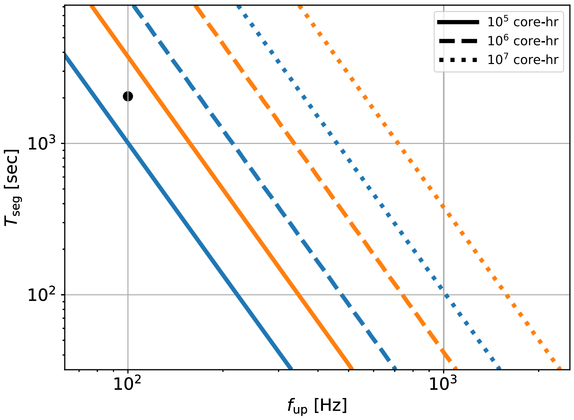

FIG. 8. Computational cost of preprocessing with various parameter values. The horizontal axis is f_{up}, and the vertical axis is T_{seg}. Blue lines and orange lines show the case of $\delta\Phi_* = 0.01$ and 0.02, respectively. Solid, dashed, and dotted lines, respectively, correspond to the contour lines of 10^5, 10^6, and 10^7 core hour. We assume a CPU Intel E5-2670 with a computational speed of 20.8 GFlops. The black dot shows our current choice of parameters $f_{\mathrm{up}} = 100$ Hz and $T_{\mathrm{seg}} = 2048$ sec.

VI. CONCLUSION

CGWs from asymmetrically rotating neutron stars or depleting boson clouds around rotating black holes are exciting targets of the ground-based interferometers. However, there are two main difficulties for all-sky searches of CGWs: (1) the computational cost due to the Doppler effect and (2) the presence of non-Gaussian line noise. In this work, we study the use of CNNs for all-sky searches when data are contaminated by line noise. We train our CNN to classify data into four classes: only Gaussian noise, astrophysical signals injected into Gaussian noise, line noise and Gaussian noise, and an astrophysical signal contaminated by line noise and Gaussian noise. Our CNN safely discriminates the presence and the absence of line noise. In the absence of a line noise, the CNN gives a false alarm probability of 0.5% and can detect an astrophysical signal with the amplitude of $\log_{10} \hat{h}_0 \gtrsim -1.64$ with 95% detection probability. On the other hand, if line noise exists in the data, the CNN's false alarm probability increases compared to the case in which line noise is absent. To remedy it, we try to modify the detection criterion. The sensitivity depth when a line is present is estimated as $\mathcal{D}^{95\%} \simeq 3.62$, with the false alarm probability of 10%. In terms of the computational time of this pipeline, the preprocess requires $O(10^{19})$ floating-point operations. It is more efficient than the standard methods, though we put the difference of the parameter range and the strain duration aside. Also, the estimated computational time for candidate selection by the CNN is $O(10^8)$ sec with a single GPU. Improving the hardware and using multiple GPUs would enable us to use CNNs in a real search.

Accounting for the conditions we neglect is necessary to apply our method to real data. In this work, we ignore the nonstationarity of detector noise, the gaps in the strain data, and the use of multiple detectors. As for line noise, we do not treat the finite coherence time and the comblike pattern. Also, we need to simulate CGWs with larger $\dot{f}$ and train CNNs to specifically handle this case. We show that our CNN is sensitive to astrophysical signals with $|\dot{f}| \lesssim 10^{-12}$ Hz/sec even if it is trained with monochromatic waveforms. On the other hand, standard all-sky search pipelines are sensitive to a signal with $|\dot{f}| \lesssim 10^{-8}$ Hz/sec. Considering the effect of $\dot{f}$ would also be useful to discriminate a line from an astrophysical signal because they have different frequency evolutions. We will extend our method to handle signals with $|\dot{f}| \gtrsim 10^{-12}$ Hz/sec in the future.

Our method includes various parameters to be optimized. The duration of a SFT segment T_{seg} is one of the crucial parameters governing the sensitivity. If we choose a short T_{seg}, the frequency resolution becomes coarse, leading to signal power being contained in one frequency bin even for large $\dot{f}$. At the same time, line noise will also stay within a frequency bin. Thus, the confusion with line noise could be serious. On the other hand, if we use a long T_{seg}, the confusion with a line noise will be suppressed; however, the range of $\dot{f}$ in which our method can be applied will become even more limited. To manage the trade-off, we need to try our method with various values of T_{seg}.

Another parameter is the residual phase $\delta\Phi_*$. If it is small, the signal after the preprocessing step can become large, resulting in better sensitivity. But, the number of the grid points in the sky, and therefore the computational cost, also increase. We should determine $\delta\Phi_*$ by considering the trade-off between the computational cost and the sensitivity. Optimizing these parameters will be done in future works.

Our systematic studies of the efficiency of CNNs to detect (quasi) monochromatic CGWs in the presence of line noise are the first of their kind. They represent a significant step toward better understanding and applying CNNs in a real all-sky search. As we show, CNNs can be used to greatly reduce the computational cost compared to existing all-sky search methods, while maintaining impressive sensitivity toward CGW signals in both the presence and absence of line noise.

ACKNOWLEDGMENTS

We thank Chris Messenger for carefully checking our draft and giving us valuable comments. We also thank Marco Cavaglia and Rodrigo Tenorio for giving us useful comments. This work was supported by JSPS KAKENHI Grants No. JP17H06358 and No. JP17H06357, A01: Testing gravity theories using gravitational waves, as a part of the innovative research area, “Gravitational wave physics and astronomy: Genesis,” and Grant No. JP20K03928. A. L. M. is a beneficiary of a FSR Incoming Postdoctoral Fellowship. This material is based upon work supported by NSF’s LIGO Laboratory which is a major facility fully funded by the National Science Foundation.

APPENDIX A: NEURAL NETWORK

An artificial neural network (ANN) is widely used in big-data analysis, e.g., image recognition and natural language processing (see [59] as a textbook). An elementary unit of an ANN is called a neuron that is inspired by neural cells in human brains. A neuron can take several values as inputs from other neurons, carry out a linear transformation, and return outputs after a nonlinear transformation called an activation function. A number of neurons are stacked into a layer, and an ANN has several layers stacked. The input data are fed into the first layer, and the output of the first layer is passed to the second layer, and so on. As a whole, the input data flow through an ANN to the last layer (the output layer). Usually, the information goes through in one direction from input to output, called forward calculation. Each layer transforms an input vector $\boldsymbol{x} \in \mathbb{R}^{N_{\mathrm{in}}}$ to an output vector $\boldsymbol{o} \in \mathbb{R}^{N_{\mathrm{out}}}$ by the transformation defined by

$$z_i = \sum_{j=1}^{N_{\mathrm{in}}} w_{ij} x_j + b_i, \qquad o_i = g(z_i). \tag{A1}$$

Here, w_{ij} and b_i are called weight and bias, respectively. The function g is an activation function. In this work, we use ReLU [65] defined by

$$g(x) = \begin{cases} x & x \geq 0, \\ 0 & x < 0. \end{cases} \tag{A2}$$

The transformation given by Eq. (A1) is often named a fully connected layer because all elements of an input vector affect every element of an output vector. It can be schematically pictured by the neurons connected by directed arrows (see Fig. 9). The number of parameters in Eq. (A1) is determined by

$$N_{\mathrm{in}} \times N_{\mathrm{out}} + N_{\mathrm{out}}, \tag{A3}$$

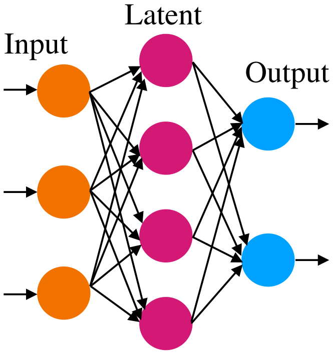

FIG. 9. Schematic picture of an artificial neural network. It consists of two fully connected layers.

where the first and second terms correspond to the size of the weight and the bias, respectively.

As stated previously, all elements in an input vector $\boldsymbol{x}$ connect to every element in an output vector $\boldsymbol{z}$ in a fully connected layer. Fukushima [66] proposed the neocognitron that has a structure in which each element of an output vector connects to a local portion of the input vector. It can be written as

$$z_i = \sum_{j=1}^{K} w_j x_{s(i-1)+j-1} + b_i. \tag{A4}$$

Here, the weight w is often referred to as a filter, and K is the size of the filter. In the transformation (A4), a small region of the input vector is convolved with a filter. The filter is gradually shifted by the width of s to cover the input vector. LeCun [67] shows that the connection given by Eq. (A4) is advantageous for extracting local patterns characterizing the input vector. Nowadays, the structure given by Eq. (A4) is referred to as a convolutional layer and is widely applied especially to image recognition tasks. Another property of convolutional layers is weight sharing. In a fully connected layer (A1), the number of weights contained in a layer is given by multiplication of the input dimension and the output dimension. It can become a tremendous number of parameters and easily stall the training. Sharing the weights between every output element can significantly reduce the number of tunable parameters and make training an ANN faster.

An input vector of a convolutional layer can be a two-dimensional tensor denoted by $x_{ai} \in \mathbb{R}^{C_{\mathrm{in}} \times N_{\mathrm{in}}}$. The index i shows an array of the data. Another index a represents a channel which corresponds to the different types of input data. For example, a color image can be characterized by three integers corresponding to the primary colors, i.e., red, green, and blue. A color picture can be represented by three datasets that have the same size as the picture and whose values determine each color's strength. The number of channels of an input is denoted by C_{in}. The weights can also be shared among different channels. Taking into account the channels, we can write the process carried out in a convolutional layer as

$$z_{ai} = \sum_{b=1}^{C_{\mathrm{in}}} \sum_{j=1}^{K} w_{abj} x_{b,s(i-1)+j-1} + b_a. \tag{A5}$$

The weights in a convolutional layer can be represented by three-dimensional tensors, $w_{abj} \in \mathbb{R}^{C_{\mathrm{in}} \times C_{\mathrm{out}} \times K}$. The bias $b_a \in \mathbb{R}^{C_{\mathrm{out}}}$ is a constant for each channel. The number of parameters can be obtained by

$$C_{\mathrm{in}} \times C_{\mathrm{out}} \times K + C_{\mathrm{out}}. \tag{A6}$$

We explained that, by virtue of the weight sharing, a convolutional layer is cheaper than a fully connected layer in terms of the number of tunable parameters. There is another way to contract the number of data points, which is called pooling. Similar to convolutional layers, pooling reduces the size of an input vector with a particular transformation. In this work, we use a max pooling defined by

$$z_{ai} = \max_{j=1,2,\ldots,K} x_{a,K(i-1)+j}. \tag{A7}$$

Typically, convolutional layers and pooling layers extract the essential features of the input data. After that, the following fully connected layers exploit the extracted features to give predictions. A neural network having convolutional layers is often called a CNN.

In our work, we apply a CNN to detect the CGWs. Detection of astrophysical signals is a typical classification problem where the classifier predicts the class to which given data likely belong. In the beginning, all tunable parameters in the neural network are initialized by assigning random values. In this state, the neural network cannot give any reliable predictions. Therefore, we need to tune all weights of the neural network (training). Here we can generate simulated training data from the signal and noise models described in Sec. II. Each simulation can be labeled by a class based on the model (e.g., Gaussian noise only, astrophysical signal injected into Gaussian noise). In other words, we *a priori* know the class where data should be classified. In general, the training with a dataset containing pairs of an input and a target value is named supervised learning. In supervised learning, the neural network is trained so that it can accurately reproduce the target data corresponding to the input data.

The output of the neural network should be appropriate for the problem we try to solve. For classification, the softmax transformation defined by

$$p_i \coloneqq \frac{\exp[x_i]}{\sum_{j=1}^{N_{\mathrm{class}}} \exp[x_j]} \tag{A8}$$

is widely used. Here, N_{class} is the number of classes. Each element p_i of the output means the probability that given data belong to the ith class. The one-hot representation (1-of-K representation) is a standard representation of the target vector for the classification problem. A target vector $\boldsymbol{t}$ represents a vector living in $\{0,1\}^{N_{\text{class}}}$. If input data belong to the ith classes, only the ith components of the target vector take a value 1. For example, if the data belong to the first class, the target vector is represented by

$$\boldsymbol{t} = (1, 0, 0, \ldots, 0). \tag{A9}$$

The vector represented as Eq. (A9) can also be interpreted in terms of the probability. That is, each element of a vector $\boldsymbol{t}$ gives a probability that the input data are in each class. For the training data, we know the class to which the data belong. Therefore, it is reasonable to assign a probability unity for the true class and zero for the rest.

In the training, the neural network's predictions and the target values need to be compared. The use of a loss function provides us with a quantitative comparison between the predictions and the targets. Depending on the problem we try to solve, we need to carefully choose a loss function. The cross entropy loss gives a distance between two probabilities. For a discrete probability, it is defined by

$$L(\boldsymbol{p}, \boldsymbol{t}) \coloneqq -\sum_{i=1}^{N_{\text{class}}} t_i \ln p_i. \tag{A10}$$

The weights of the neural network are optimized so that the expected value of the loss function over the dataset is minimized. We cannot analytically optimize the weights; therefore, we need an iterative scheme to obtain better weights. Schematically, an optimization scheme can be written as

$$w' = w - \eta \frac{\partial L}{\partial w}, \tag{A11}$$

where w is a weight in the neural network, w' is an updated value of the weight, and η is the so-called learning rate and governs how sensitive the updated amount is to the gradients of the loss function. During the training, the neural network is gradually optimized by repeating a set of processes to (1) feed input data to the ANN, (2) return the prediction, (3) evaluate the loss function and its gradients, and (4) update the weights.

Because of a large number of tunable parameters, neural networks sometimes learn to only fit the training data. If that happens, neural networks do not have high accuracy on new data. Such a phenomenon is called overfitting. To check whether overfitting occurs, we prepare another dataset (validation data) and monitor the loss of the validation data in each epoch. If the neural network overfits the training data, the validation loss gradually deviates from the training loss and can increase. Figure 10 shows the training and validation loss as a function of the training epoch. We find that the loss in our CNN tends to converge well, and the validation loss follows the training loss. Thus, overfitting did not happen during training, and we use the last state of the CNN for testing.

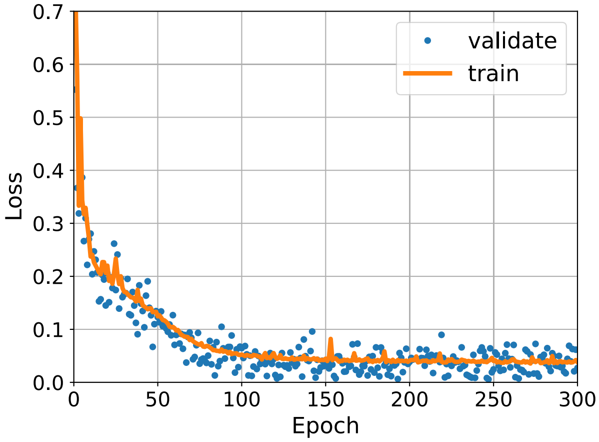

FIG. 10. Training curve of CNNs of four classes. Orange line shows training loss. Blue dots indicate validate losses. The validation loss has a larger variance than the training loss. This is because of the difference of the number of the training data and the validation data.

APPENDIX B: FINE-TUNING OF CNN

In general, it takes much time to train a neural network from scratch. Therefore, as a first step, a neural network is trained for a more manageable problem than the one we try to solve finally. This step is called pretraining. Then, a pretrained neural network is optimized for the problem we want to solve. The technique optimizing a neural network in a hierarchical manner is called fine-tuning. In this appendix, our CNN is fine-tuned, including $\dot{f}$.

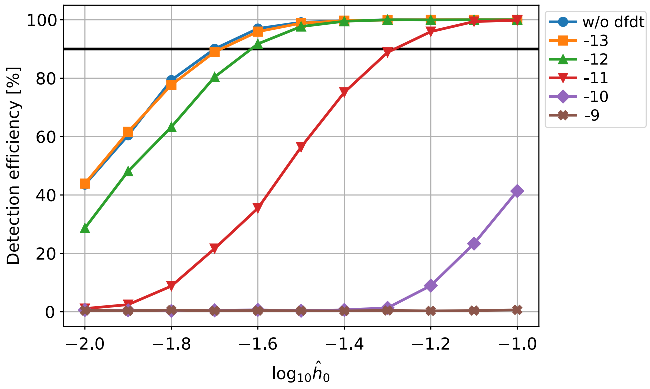

FIG. 11. Detection probabilities for astrophysical signals with various values of $\dot{f}$. We set $\dot{f}$ from -1.0×10^{-13} to -1.0×10^{-9} Hz/ sec.

As we explained, for $|\dot{f}| \gtrsim 10^{-11}$ Hz/ sec, the signal power would dissipate to several frequency bins. Therefore, we guess that it is useless to include the signal with $|\dot{f}| \gtrsim 10^{-11}$ Hz/ sec in the training data. The dataset is generated with the same setup as shown in Sec. III except that $\dot{f}$ is randomly sampled from $10^{-14} \leq |\dot{f}| \leq 10^{-11}$ Hz/ sec. We use the trained CNN as an initial state, set the learning rate to 10^{-4}, and update the weights of the fully connected layers for 150 epochs with the frozen convolutional layers. Figure 11 shows the detection probabilities for signals with various $\dot{f}$. For $|\dot{f}| \leq 10^{-11}$ Hz/ sec, the detection probabilities get improved by the fine-tune. It does not improve for $|\dot{f}| = 10^{-9}$ Hz/ sec, but it is a predictable result because the dataset for fine-tuning does not include the signal with $\dot{f} = 10^{9}$ Hz/ sec.

[1] P. D. Lasky, Pub. Astron. Soc. Aust. **32**, e034 (2015).

[2] K. Glampedakis and L. Gualtieri, Astrophysics and Space Science Library **457**, 673 (2018).

[3] K. Riles, Mod. Phys. Lett. A **32**, 1730035 (2017).

[4] M. Sieniawska and M. Bejger, Universe **5**, 217 (2019).

[5] R. Tenorio, D. Keitel, and A. M. Sintes, Universe **7**, 474 (2021).

[6] D. R. Lorimer and M. Kramer, *Handbook of Pulsar Astronomy* (Cambridge University Press, Cambridge, 2004), Vol. 4.

[7] http://www.atnf.csiro. au/people/pulsar/psrcat/.

[8] R. N. Manchester, G. B. Hobbs, A. Teoh, and M. Hobbs, Astron. J. **129**, 1993 (2005).

[9] A. Arvanitaki, S. Dimopoulos, S. Dubovsky, N. Kaloper, and J. March-Russell, Phys. Rev. D **81**, 123530 (2010).

[10] R. Brito, V. Cardoso, and P. Pani, Lect. Notes Phys. **906**, 1 (2015).

[11] S. D'Antonio *et al.*, Phys. Rev. D **98**, 103017 (2018).

[12] M. Isi, L. Sun, R. Brito, and A. Melatos, Phys. Rev. D **99**, 084042 (2019).

[13] C. Palomba *et al.*, Phys. Rev. Lett. **123**, 171101 (2019).

[14] A. L. Miller, S. Clesse, F. De Lillo, G. Bruno, A. Depasse, and A. Tanasijczuk, Phys. Dark Universe **32**, 100836 (2021).

[15] A. L. Miller, N. Aggarwal, S. Clesse, and F. De Lillo, Phys. Rev. D **105**, 062008 (2022).

[16] O. Pujolas, V. Vaskonen, and H. Veermäe, Phys. Rev. D **104**, 083521 (2021).

[17] H. Guo and A. Miller, arXiv:2205.10359.

[18] P. Jaranowski, A. Krolak, and B. F. Schutz, Phys. Rev. D **58**, 063001 (1998).

[19] P. Astone, A. Colla, S. D'Antonio, S. Frasca, and C. Palomba, Phys. Rev. D **90**, 042002 (2014).

[20] B. Krishnan, A. M. Sintes, M. A. Papa, B. F. Schutz, S. Frasca, and C. Palomba, Phys. Rev. D **70**, 082001 (2004).

[21] J. Bayley, G. Woan, and C. Messenger, Phys. Rev. D **100**, 023006 (2019).

[22] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. D **96**, 062002 (2017).

[23] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. D **96**, 122004 (2017).

[24] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. D **97**, 102003 (2018).

[25] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. D **100**, 024004 (2019).

[26] R. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. D **103**, 064017 (2021).

[27] R. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Phys. Rev. D **105**, 082005 (2022).

[28] R. Abbott *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), Phys. Rev. D **105**, 102001 (2022).

[29] R. Abbott *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), Astrophys. J. **932**, 133 (2022).

[30] R. Abbott *et al.* (LIGO Scientific, Virgo, and KAGRA Collaborations), arXiv:2201.00697.

[31] P. B. Covas *et al.* (LSC Collaboration), Phys. Rev. D **97**, 082002 (2018).

[32] E. Cuoco *et al.*, Mach. Learn. Sci. Technol. **2**, 011002 (2021).

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

[34] D. George and E. A. Huerta, Phys. Lett. B **778**, 64 (2018).

[35] R. E. Colgan, K. R. Corley, Y. Lau, I. Bartos, J. N. Wright, Z. Marka, and S. Marka, Phys. Rev. D **101**, 102003 (2020).

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

[37] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, Nat. Phys. **18**, 112 (2022).

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

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

[40] M. Dax, S. R. Green, J. Gair, J. H. Macke, A. Buonanno, and B. Schölkopf, Phys. Rev. Lett. **127**, 241103 (2021).

[41] H.-S. Kuo and F.-L. Lin, Phys. Rev. D **105**, 044016 (2022).

[42] H. Nakano, T. Narikawa, K.-i. Oohara, K. Sakai, H.-a. Shinkai, H. Takahashi, T. Tanaka, N. Uchikata, S. Yamamoto, and T. S. Yamamoto, Phys. Rev. D **99**, 124032 (2019).

[43] H. Shen, E. A. Huerta, E. O'Shea, P. Kumar, and Z. Zhao, Mach. Learn. Sci. Technol. **3**, 015007 (2022).

[44] T. S. Yamamoto and T. Tanaka, arXiv:2002.12095.

[45] S. Bhagwat and C. Pacilio, Phys. Rev. D **104**, 024030 (2021).

[46] A. L. Miller *et al.*, Phys. Rev. D **100**, 062005 (2019).

[47] A. Miller *et al.*, Phys. Rev. D **98**, 102004 (2018).

[48] G. Morrás, J. García-Bellido, and S. Nesseris, Phys. Dark Universe **35**, 100932 (2022).

[49] D. Chatterjee, S. Ghosh, P. R. Brady, S. J. Kapadia, A. L. Miller, S. Nissanke, and F. Pannarale, Astrophys. J. **896**, 54 (2020).

[50] A. Mytidis, A. A. Panagopoulos, O. P. Panagopoulos, A. Miller, and B. Whiting, Phys. Rev. D **99**, 024024 (2019).

[51] C. Dreissigacker, R. Sharma, C. Messenger, R. Zhao, and R. Prix, Phys. Rev. D **100**, 044009 (2019).

[52] F. Morawski, M. Bejger, and P. Ciecieląg, Mach. Learn. Sci. Technol. **1**, 025016 (2020).

[53] B. Beheshtipour and M. A. Papa, Phys. Rev. D **101**, 064009 (2020).

[54] J. Bayley, C. Messenger, and G. Woan, Phys. Rev. D **102**, 083024 (2020).

[55] S. Walsh *et al.*, Phys. Rev. D **94**, 124010 (2016).

[56] T. S. Yamamoto and T. Tanaka, Phys. Rev. D **103**, 084049 (2021).

[57] P. Jaranowski and A. Krolak, *Analysis of Gravitational-Wave Data* (Cambridge University Press, Cambridge, England, 2009).

[58] J. Aasi *et al.* (LIGO Scientific Collaboration), Classical Quantum Gravity **32**, 074001 (2015).

[59] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).

[60] H. Nakano, H. Takahashi, H. Tagoshi, and M. Sasaki, Phys. Rev. D **68**, 102003 (2003).

[61] The Tukey window would reduce both the powers of signal and noise. Therefore, Eq. (2.43) overestimates the variance of noise. Because the CGW signal and the line noise are generated with considering the Tukey window, the SNR of simulated data could be underestimated. In this sense, our estimation of detection efficiency is conservative.

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

[63] A. Paszke *et al.*, *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019).

[64] Assuming the use of ten GPUs that are twice faster than the GTX1080Ti that is used in this work, we have computational time $T_{\mathrm{CNN}} \simeq 5.0 \times 10^6$ [sec].

[65] V. Nair and G. E. Hinton, in *Proceedings of the 27th ICML* (Omnipress, Madison, WI, 2010), pp. 807–814.

[66] K. Fukushima, Biol. Cybern. **36**, 193 (1980).

[67] Y. LeCun, in *Connectionism in Perspective*, edited by R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels (Elsevier, Zurich, 1989). An extended version was published as a technical report of the University of Toronto.