# Improved deep learning techniques in gravitational-wave data analysis

Heming Xia,[1] Lijing Shao,[2,3,*] Junjie Zhao,[4] and Zhoujian Cao[5]

[1]*Department of Astronomy, School of Physics, Peking University, Beijing 100871, China*
[2]*Kavli Institute for Astronomy and Astrophysics, Peking University, Beijing 100871, China*
[3]*National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100012, China*
[4]*School of Physics and State Key Laboratory of Nuclear Physics and Technology, Peking University, Beijing 100871, China*
[5]*Department of Astronomy, Beijing Normal University, Beijing 100875, China*

(Received 9 November 2020; accepted 23 December 2020; published 21 January 2021)

In recent years, convolutional neural network (CNN) and other deep learning models have been gradually introduced into the area of gravitational-wave (GW) data processing. Compared with the traditional matched-filtering techniques, CNN has significant advantages in efficiency in GW signal detection tasks. In addition, matched-filtering techniques are based on the template bank of the existing theoretical waveform, which makes it difficult to find GW signals beyond theoretical expectation. In this paper, based on the task of GW detection of binary black holes, we introduce the optimization techniques of deep learning, such as batch normalization and dropout, to CNN models. Detailed studies of model performance are carried out. Through this study, we recommend to use batch normalization and dropout techniques in CNN models in GW signal detection tasks. Furthermore, we investigate the generalization ability of CNN models on different parameter ranges of GW signals. We point out that CNN models are robust to the variation of the parameter range of the GW waveform. This is a major advantage of deep learning models over matched-filtering techniques.

DOI: 10.1103/PhysRevD.103.024040

## I. INTRODUCTION

On February 11, 2016, LIGO and Virgo Collaboration announced the first detection of gravitational-wave (GW) signals from September 14, 2015, the so-called GW150914 [1–3]. The detected GW signal comes from a binary black hole (BBH) merger. The masses of the BBH are estimated to be 29 $M_\odot$ and 36 $M_\odot$. The successful observation of GW signals has provided valuable experimental data for GW astronomy, setting off a wave of GW researches. So far, 50 GW signals from compact binary coalescences have been successfully detected [4–10]. Except for the binary neutron star (BNS) merger event, GW170817 [11] and three other *strictly speaking* unclear merger events—GW190814 [7], GW190425 [8], and GW190426_152155 [10]—the remaining 46 signals all come from BBH mergers.

Currently, both LIGO and Virgo mainly use the matched-filtering method [12–15] to detect GW signals. This method builds a theoretical waveform template bank to match the monitored data and captures trigger signals as candidates for further verification [12]. Matched filtering plays a vital role in the processing of GW signal detection. However, it has shortcomings that should not be overlooked [16,17]. Matched-filtering method requires a full search in the template bank to match the signal, which limits the data processing speed and, if a big template bank is used, has difficulty to meet the needs of real-time observation. In addition, with continuous expansion of theoretical waveforms in an enlarging parameter space, the search space of matched filtering increases, which leads to an increase of data processing time and a reduction in the processing speed [17].

In recent years, many machine learning methods have been developed in GW signal detection tasks [18–23]. In the machine learning field, as AlexNet won the championship in the ImageNet competition in 2012 [24], deep learning algorithms stood out and achieved great success in many fields such as image classification, natural language processing, and speech recognition [25,26]. In terms of classification tasks, compared with traditional machine learning algorithms, many deep learning algorithms, including convolutional neural networks (CNNs), have made significant progress in model accuracy and model complexity. Besides, the characteristics of the deep learning algorithm make the time-consuming training process be completed offline before the actual data analysis. It greatly reduces the amount of calculation in the online process and meets the need of real-time detection [26].

At present, deep learning methods, especially CNNs, have been widely explored in GW data processing [27–35].

[*]lshao@pku.edu.cn

In 2017, George and Huerta [27] firstly applied CNN to GW signal detection tasks. They generated mock GW signals from BBH mergers and added them into white Gaussian noise to generate simulation data sets. They pointed out that the sensitivity, namely, the fraction of signals which are correctly identified, of CNNs is similar to the matched-filtering method, while its speed has been greatly improved. The work in the same period by Gabbard *et al.* [28] compared the false alarm rate and the receiver operating characteristic (ROC) curve between CNN models and the matched-filtering method, leading to a similar conclusion. Subsequently, the application of deep learning in the field of GW signal detection has been expanded greatly. Krastev [29] indicates that deep learning models work better on BNS mergers than BBH mergers. More deep learning models such as residual network, fully CNN, and other structures have been introduced [34–39], and the research field has been continuously expanding. Variational autoencoders and Bayesian neural networks are used for parameter estimation of GW signals [30,31]. Long short-term memory network has made progress in the field of GW signal noise reduction, which proves that it can effectively remove environmental noise and restore the GW signal under noise [32]. In the sky localization searching task of GW signals, deep learning methods such as CNNs have also achieved good results [33].

However, almost all deep learning algorithms such as CNNs used in the current researches are basic models. It means that they can be further optimized. In addition, many studies have pointed out that deep learning models can maintain a certain degree of robustness to GW signals beyond the range of the training set parameters [27,28,35]. However, there is no specific research on this aspect. Grounded on the above two points, we conduct experiments on the optimization effects of several deep learning techniques in the field of GW signal detection. The result shows that, compared with the basic model, the model with improved techniques achieves better performance. On the low signal-to-noise ratio (SNR) data set, the model with multiple improved techniques has an accuracy rate of 84% on the testing set, 12% higher than that of the basic model, and an area under curve (AUC) score of 0.91, 6% higher than that of the basic model. On the overall data set, our model with multiple improved techniques has an accuracy rate of 94% on the testing set, 4% higher than that of the basic model, and an AUC score of 0.98, 2% higher than that of the basic model. Moreover, we make a detailed research on the robustness of CNN models on GW signal detection tasks. Our experiments show that the CNN model has good robustness for data of different parameter ranges for masses and spins.

This paper is organized as follows. In Sec. II, we give a brief overview on deep learning. In Sec. III, we introduce our simulated data set to be used in our experiments. Then the improved techniques for CNN and the corresponding experimental results are shown in Secs. IV and V, respectively. In Sec. VI, we investigate the generalization ability of the CNN model in different parameter ranges.

## II. DEEP LEARNING

Traditional machine learning methods include $k$-nearest neighbor, decision tree, support vector machine, and so on [40]. The advantage of machine learning methods is that to some extent, they can replace the process of human learning. Through training on a large data set, these models can learn the *relationship* between data so that to classify, predict, and help human to make decisions [41]. However, when traditional machine learning methods are applied to specific tasks, they usually have difficulty processing the original data. In some cases, researchers have to manually extract data features and put them into algorithms. Besides, traditional machine learning methods are usually limited by their fixed model structure. It is difficult for these algorithms to achieve rapid improvement in computing power and accuracy [26].

Deep learning overcomes some limitations of traditional machine learning methods. Its algorithm is derived from neural network which is a subfield of traditional machine learning. This algorithm solves the limitation of the depth of neural network and increases the computational power of the model. So far, many deep learning models such as CNN, residual network, and long and short time memory have been proposed. At present, deep learning has achieved substantial success in face recognition, automatic driving, speech processing, and many other fields [26]. For the sake of a self-contained work, below we briefly review the principle of deep learning, including the structure of neurons, the basic principle of neural network, and the model of CNN.

### A. Neuron

The idea of neural networks in machine learning evolved from biological models. Generally speaking, neural network is a network of parallel interconnections composed of simple adaptive units [42]. Its organization can simulate the interaction of the biological nervous system to real-world objects. The neuron is the "simple unit" in the above definition. In 1943, McCulloch and Pitts [43] abstracted it into the simple model shown in Fig. 1, namely, the "M-P neuron model." In this model, each neuron receives input data $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ from previous neurons. The input is multiplied by the weight $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$, plus the bias $b$, and then is injected into the activation function $\sigma$ to obtain the output $y$. A single neuron can be represented by the following formula using vectors:

$$y = \sigma(\boldsymbol{w}\boldsymbol{x}^{\intercal} + b). \tag{1}$$

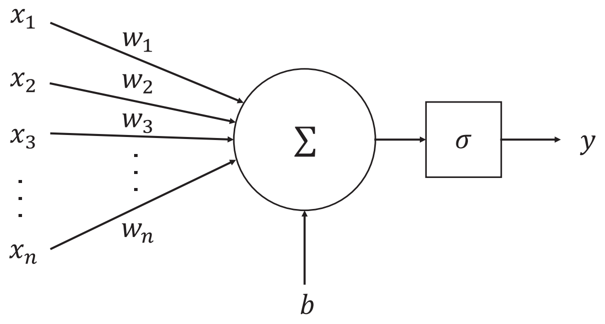


FIG. 1. Architecture of a neuron.

The activation function $\sigma$ introduces nonlinear operations into neurons. Otherwise, the structure of neurons will simply be a superposition of linear operations, and the power of the network will be greatly reduced. Activation function $\sigma$ comes in many forms [44]. In this paper, we use the most commonly used activation function RELU [45] in our study,

$$\sigma(x) = \begin{cases} x, & x > 0, \\ 0, & x \leq 0. \end{cases} \tag{2}$$

In general, the current neuron will only output positive values after calculating the weighted sum of data from the first $N$ neurons. From the feature level, it can be understood as the following. After a linear combination of $N$ features, neurons input the combined features into the activation function to obtain the output features.

## B. Neural network

The simplest example of neural network is fully connected neural network (FCNN). The structure of FCNN is shown in Fig. 2. A manually specified number of neurons

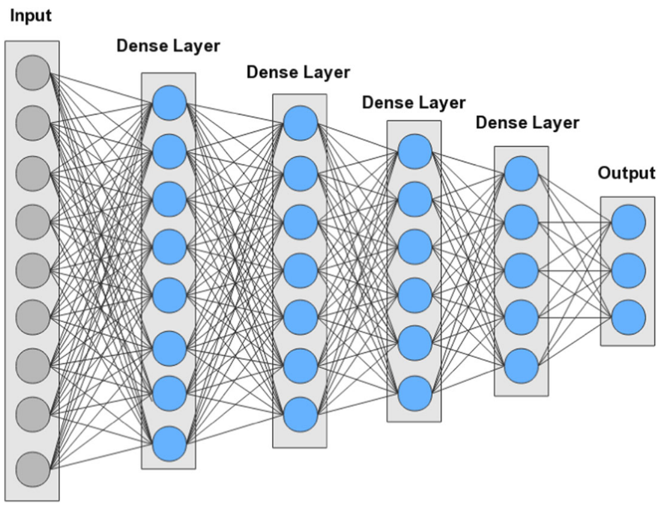


FIG. 2. Structure of a FCNN.

constitute each layer, which is called a "dense layer" of the network. Neurons between different layers are independent. Each neuron receives data from the former layer, calculates the result, and puts them into all neurons of the next layer [46]. Note that the input data size has a linear relationship with the parameters of the first layer in FCNN [26]. With the input data size increases, the number of parameters in the network will increase correspondingly, slowing down the learning speed of the model and increasing the requirement of data storage [25].

Neural network is a supervised learning algorithm, whose characteristic is to use the labeled data—which is the correct output $y$—for training, and test the model on unlabeled data. A loss function is defined to measure the difference between the model output $\hat{y}$ and the correct $y$. The expectation for training is to make the value of loss function as small as possible [26]. The parameters to be informed in the neural network are the weights $\boldsymbol{w}$'s and biases $b$'s in each layer of neurons. According to the gradient descent strategy, parameters in the neural network are updated to the direction where the value of loss function decreases [47]. The gradient descent strategy reads

$$\theta^{\mathrm{new}} = \theta^{\mathrm{old}} - \alpha \nabla_{\theta} J(\theta), \tag{3}$$

where $\theta$ is the parameter to be updated, $J$ is the loss function, and $\alpha$ is the manually specified learning rate which controls the speed of parameter updates.

The learning of neural network is based on the training data set collected and annotated by human beings. However, when we finally apply the model to real tasks, we hope the neural network to have good *generalization ability.* Generalization ability with respect to the neural network is defined as the ability of the network to handle unseen patterns [48]. In other words, this concept measures how accurately an algorithm is able to predict outcome values for previously unseen data [49]. The testing set is a good type of unseen data. The data characteristics of the testing set are similar to the training set. In the meantime, its distribution is independent of the training set, and it does not appear in the training process [26]. Therefore, the final result of the testing set is a reliable index to evaluate the performance of the neural network.

However, if the number of parameters in the network is more compared to the samples in the training set, *overfitting* occurs [50]. If there are too many parameters in the neural network, the predicting power of the model will get too strong, which makes the model fit the (noisy) characteristics of the training data set too much in the learning process. As a result, the model is only effective for the data samples that appear in the training set, resulting in the so-called the overfitting problem [51]. Therefore, how to design the model structure of neural network with appropriate layers and neuron numbers with strong data generalization ability is an outstanding issue.

### C. Convolutional neural network

As shown in Fig. 3, the structure of CNN is divided into the convolutional layer, the pooling layer, and the dense layer [26]. Each convolutional layer is composed of a specified number of kernels. Each kernel multiplies the input feature values with weights and adds the biases to obtain outputs. Different kernels get different parameter values after training. The pooling layer itself does not contain any parameters. Take the max-pooling layer as an example. After receiving the input data, this layer scans the data according to a specified stride within a window of a certain length. Then, it outputs the maximum value of the data in each scanning window [52]. Therefore, the pooling layer compresses the data. It checks all the features in the scanning window and chooses the most important one [53]. There are other pooling methods, e.g., the average pooling, which outputs the average value of the data in the window [54].

The pooling layer plays an important role in improving the receptive field of CNN. After the data pass through the pooling layer, the original data length gets shortened, and the kernel in the next convolutional layer can handle a larger range of data than the convolution window in the previous layer, thereby it expands the convolutional layer's overall operation range of the data [24]. After extracting the features of the input data through convolutional layers and pooling layers, the flattened feature map will be put into the fully connected layer to obtain the final output [26].

Since the parameters to be learned by the convolutional layer are only the parameter values in the convolution kernel, they are independent of the input data size. Therefore, CNN reduces the number of free parameters, allowing the network to be deeper with fewer parameters [54]. Due to its unique model structure and powerful data processing capability, CNN is being widely used [26].

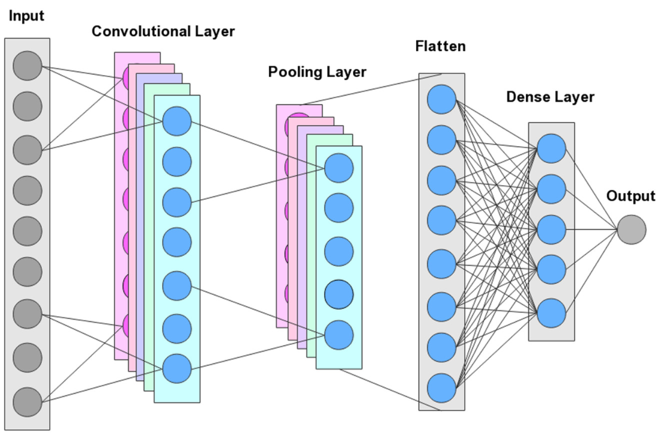


FIG. 3. Structure of a CNN.

## III. SIMULATED DATA SET

In this section, we discuss our strategy to simulate GW data and build different data sets for machine learning studies.

### A. Data obtaining

Usually, GW waveforms are divided into three stages: inspiral, merger, and ringdown. The signal detected by a single GW detector is

$$h(t) = F^{+}(t)h_{+}(t) + F^{\times}(t)h_{\times}(t), \tag{4}$$

where $h_{+}$ and $h_{\times}$ are two polarization modes of GWs, $F^{+}$ and $F^{\times}$ are the corresponding pattern functions of these two polarization modes as functions of the sky localization and the polarization angle [55].

In this work, we focus on GW signals generated by BBH mergers. We use the effective-one-body numerical-relatively model with aligned spins [56] to simulate the waveform. Without losing generality, in addition to the SNR of GW signals, we focus on intrinsic parameters, i.e., masses and spins. The extrinsic parameters, such as the polarization angle and the sky localization, are all fixed to fiducial values. The possible precession effect of BBHs is not considered, since we are using the aligned-spin waveform. The spin parameter is denoted by $\chi$. We set the luminosity distance $D_{\rm L} = 100$ Mpc and neglect the redshift effect of the GW signal. Such a setting makes us focus on the machine learning algorithm, and the assumptions can easily be relaxed when needed. Worth to mention that, in practice we have tested the effects brought by the inclusion of extrinsic parameters, such as sky location of the GW source, inclination of the BBH orbit, and the polarization angle of the GW. Consistent optimization effects were obtained with extrinsic parameters. Because the dependence of the GW waveform on extrinsic parameters is much simpler than that of intrinsic parameters for GWs from spin-aligned BBHs, in the following we will focus on the intrinsic parameters. It is straightforward to augment with extrinsic parameters in our machine learning data analysis.

We use the open-source tool provided by Gebhard *et al.* [35] to generate data. This tool generates GW signals based on PyCBC [57,58] and LALSuite [59] platforms. With given parameters, analog signals from LALSuite contain two time series, i.e., the two polarization modes of GW signals. The above sequences are combined with the corresponding antenna functions $F^{+,\times}$ according to Eq. (4). The signal offset caused by the distance difference between LIGO's Hanford and Livingston detectors is properly introduced.

The final GW signal sequence used is

$$s(t) = h(t) + n(t), \tag{5}$$

where $h(t)$ is the GW waveform obtained by the above simulation (4), and $n(t)$ is the detector's noise to the strain. The advanced LIGO's (aLIGO's) power spectral density at the "zero-detuned high-power" design sensitivity (aLIGOZeroDetHighPower) [60] is used to simulate the Gaussian white noise. After inserting the analog waveform into the noise, we can calculate the SNR of the strain. A rescaling of it, corresponding to a rescaling in the distance, can achieve other desired SNR values [55]. Finally, we get the GW strain with specific SNR values. The strain needs to be preprocessed before being used as the final data, which is similar to PyCBC's GW data processing. Preprocessing stage includes two steps. The first one is data whitening. The aLIGO's design sensitivity is used to whiten the original strain and to filter out the spectral components of the environmental noise so as to properly scale its influence on the strain. The second step is filtering. We filter out the frequency components below 20 Hz to eliminate the influence of the Newtonian noise in the low frequency.

## B. Data set building

This work involves experiments on multiple data sets which have the same structure. They have the following characteristics:

(i) with the given parameter range, the data parameters, masses $m_{1,2}$, and spins $\chi_{1,2}$, are all in the form of random sampling;
(ii) the ratio of the samples containing the GW signal and pure noise in the data set is 1 to 1;
(iii) the duration of each sample is 2 seconds, and the sampling rate is 2048 Hz, that is, each sample is a time series with a length of 4096;
(iv) considering the symmetry of mass parameters in a BBH, we use $m_1 \geq m_2$ convention to sample masses;
(v) the time-series data of the samples all use the Hanford detector sequence (the H1 sequence);
(vi) in this work, we do not consider the influence of sky position of the GW signal in the strain, that is to say, the peak value of the GW signal is located in the same position of the strain in the datasets. These specifics are natural for a study of such kind.

TABLE I. Parameters for training data sets.

| Data set | $m_1$ $(M_\odot)$ | $m_2$ $(M_\odot)$ | $\chi_{1,2}$ | SNR |
|---|---|---|---|---|
| 1.1 | [10, 80] | [10, 80] | 0 | [7, 7.5] |
| 1.2 | [10, 80] | [10, 80] | 0 | [7, 15] |
| 2.1 | [30, 60] | [30, 60] | 0 | 8 |
| 3.1 | 30 | 45 | $[-0.5, 0.5]$ | 8 |
| 4.1 | [10, 80] | [10, 80] | $[-0.998, 0.998]$ | [7, 15] |

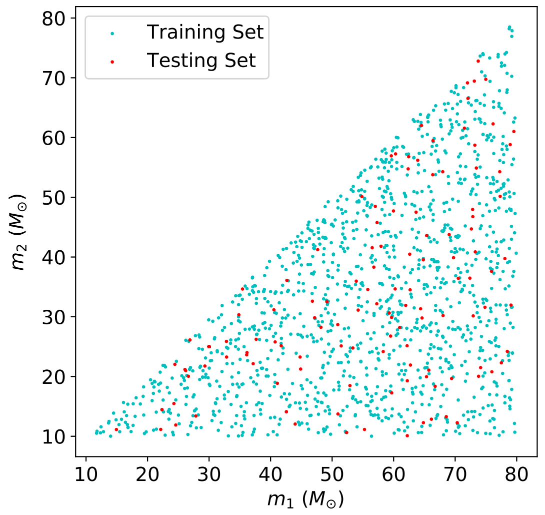
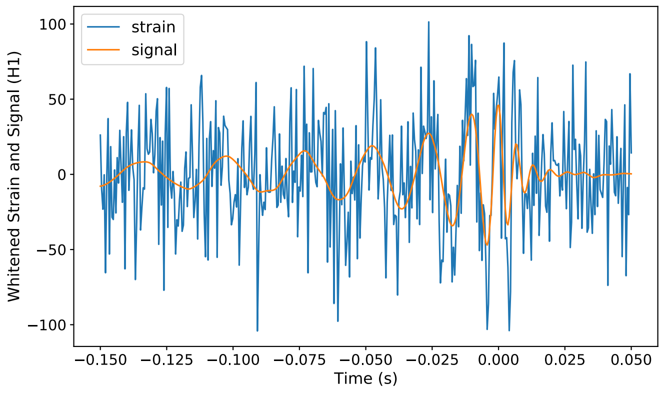


FIG. 4. Mass distribution in data set 1.1. We use $m_1 \geq m_2$ as a convention to randomly sample mass parameters. Cyan points are the training set samples and red points are the testing set samples.

Totally, we construct five data sets for training and five data sets for testing, which are annotated as *training data sets* and *testing data sets* hereafter.



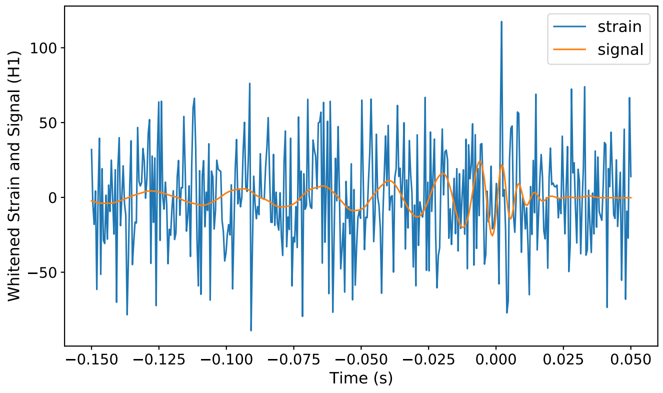


FIG. 5. Examples for the simulated GW strain: $(m_1, m_2) = (75.61\ M_\odot, 18.64\ M_\odot)$ and $\rho = 14.56$ (upper); $(m_1, m_2) = (67.98\ M_\odot, 27.73\ M_\odot)$ and $\rho = 7.41$ (lower). The orange line is the normalized GW signal in the sample strain.

TABLE II. Parameters for testing data sets.

| Data set | $m_1$ $(M_\odot)$ | $m_2$ $(M_\odot)$ | $\chi_{1,2}$ | SNR |
|---|---|---|---|---|
| 1 | [10, 80] | [10, 80] | 0 | [7, 0.5, 15] |
| 2 | [10, 10, 80] | [10, 10, 80] | 0 | 8 |
| 3 | 30 | 45 | $[-0.998, 0.25, 0.998]$ | 8 |
| 4 | [10, 80] | [10, 80] | $[-0.998, 0.998]$ | [7, 15] |
| 5 | [10, 80] | [10, 80] | $[-0.998, 0.998]$ | [7, 0.5, 15] |

(i) Training data sets: Each training data set is consisted of a training set and a testing set. The training set and the testing set contain 5000 and 500 samples, respectively. When we train the model on each data set, the model is first trained on the training set and then tested on the testing set.

(ii) Testing data sets: Testing data sets contain many subtesting sets (annotated as subdatasets below) to test the model performance on different parameter ranges. Each subdataset contains 500 samples.

The parameters for training data sets used in this work are shown in Table I. Taking data set 1.1 as an example. This data set only considers three parameters: component masses of the BBHs, $m_1$ and $m_2$, and the SNR $\rho$. The spin parameters of the BBHs are set to zero. The mass range of both black holes is $m_i \in [10\ M_\odot, 80\ M_\odot]$ $(i = 1, 2)$, and the range of SNR is $\rho \in [7, 7.5]$. The training set has 5000 samples in terms of physical parameters, including 2500 samples which contain the simulated GW signal and 2500 samples of pure noise. The testing set contains 500 samples, including 250 samples containing the simulated GW signal and 250 pure-noise samples. The mass parameter of each sample is randomly drawn from a given range ($m_1 \geq m_2$ as defaulted), whose distribution is shown in Fig. 4. Figure 5 gives two GW examples from data set 1.2 with a large SNR (upper panel) and a small SNR (lower panel).

The parameter settings of testing data sets are shown in Table II. As shown above, the parameters used to construct each subdataset are given in the $[\mathtt{min}, \mathtt{step}, \mathtt{max}]$ format, where `min` and `max` define the overall sampling range of the parameter, and each subdataset is constructed with a uniform *step* size. Take the testing data set 1 as an example, which consists of 16 subdatasets. The SNR parameter sampling range of the first subdataset is [7, 7.5]. Each parameter is randomly sampled within this range, and the number of samples is 500. The testing data set will be used to investigate the detection ability of the model in different parameter intervals and explore model robustness with the change of parameter values.

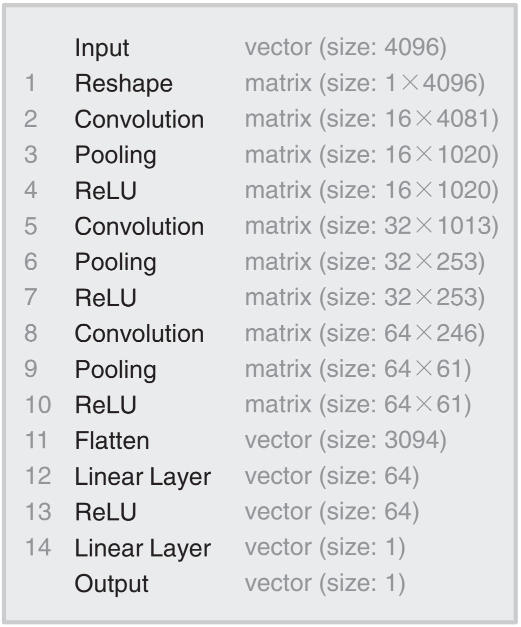
## IV. IMPROVING TECHNIQUES

### A. Model baseline

After preliminary experiments with parameters on networks with different depths and hyperparameter values, we decide to use the network structure similar to that of George and Huerta [27] as the baseline CNN model. Its structure is shown in Fig. 6.

The model receives GW strain as input and outputs the discriminant value of classification. In the model, the classification threshold is set to 0.5, that is, a sample with a discriminant value greater than 0.5 is judged as a positive sample (i.e., containing a GW signal); otherwise, it is judged as a negative one (i.e., pure noise). The baseline model contains three convolutional layers. The number of channels of each convolutional layer is 16, 32, and 64, and the size of the convolution kernel for them is 16, 8, and 8, respectively. After each convolutional layer, a pooling layer and an activation layer are provided. The pooling layer uses maximum pooling with a pooling window of 4 and a step size of 4, which means that the feature sequence is down-sampled to a quarter of the original sequence length and retains the feature with the largest value. The active layer uses the RELU function shown in Eq. (2). Subsequently, the feature sequences extracted by the convolution structure are input to the fully connected layer to achieve discriminative classification. The model outputs discriminant values to determine whether the sequence contains a GW signal.

In this work, cross entropy is used as the loss function to update the gradient. Cross entropy is an important concept proposed by Shannon [61] in information theory. It is often used to measure the difference between the predicted distribution and the true distribution [61]. Let the label of each sample be $y$ and the discriminant value given by the model be $p$. The cross entropy is



| | | |
|---|---|---|
| | Input | vector (size: 4096) |
| 1 | Reshape | matrix (size: 1×4096) |
| 2 | Convolution | matrix (size: 16×4081) |
| 3 | Pooling | matrix (size: 16×1020) |
| 4 | ReLU | matrix (size: 16×1020) |
| 5 | Convolution | matrix (size: 32×1013) |
| 6 | Pooling | matrix (size: 32×253) |
| 7 | ReLU | matrix (size: 32×253) |
| 8 | Convolution | matrix (size: 64×246) |
| 9 | Pooling | matrix (size: 64×61) |
| 10 | ReLU | matrix (size: 64×61) |
| 11 | Flatten | vector (size: 3094) |
| 12 | Linear Layer | vector (size: 64) |
| 13 | ReLU | vector (size: 64) |
| 14 | Linear Layer | vector (size: 1) |
| | Output | vector (size: 1) |

FIG. 6. Basic CNN model architecture that is used in this work [27].

$$J = -(y \cdot \log p + (1 - y) \cdot \log(1 - p)). \tag{6}$$

We use ADAM [62] as a gradient descent strategy to update our model parameters. This strategy inputs the samples into the model in batches to calculate the gradient and updates the parameters. After trying batch number values of 5, 10, 25, 50, 100, 200, 250, and 500, we take the value 25, on which the model has the highest accuracy. The learning rate is set to $5 \times 10^{-5}$, and the training rate is reduced 10 times every 20 epochs to avoid overfitting the model.

The code implementation of our work is based on the PYTORCH framework [63], which uses the CUDA deep learning library [64] to accelerate the GPU's model operation. Our work deploys experiments on NVIDIA TITAN X GPU.

### B. Improving techniques

We now discuss several improvements—dropout, batch normalization, and the $1 \times 1$ convolution—that we experiment on the model of George and Huerta [27].

Dropout was first proposed by Hinton *et al.* [65] in 2012 to improve the overfitting problem of neural networks. Subsequently, dropout has become one of the widely used techniques in deep learning [66]. Its basic idea is that, in each batch of training, the probability $p$ is artificially specified, so that the neurons in the fully connected layer stop working with the probability $p$, and their parameter values are set to zero.

The advantages of dropout are the following [65]. First, in the training process of each batch, because the neurons of each layer are inactivated with probability $p$, the network structure of each training is different. The overall model training is equivalent to the joint decision-making process between multiple neural networks with different structures, which helps to improve the problem of overfitting. Second, dropout results in that, two neurons do not necessarily appear in the same network structure each time so that the parameter update no longer depends on the joint decision of some neurons with fixed relationships. At the feature level, this technique prevents decision making from overdependence on certain features and forces the model to learn more robust feature representations [26].

Batch normalization is a neural network training technique proposed by Ioffe and Szegedy [67] in 2015. Its specific idea is the following. In the training process of each batch, after the data pass through the activation layer, the activation value of each batch of data is normalized. That is, the average value of the sample data of each batch is normalized to 0, and the variance is normalized to 1. In a batch of data of length $m$, the activated data are set to $\mathcal{B} = \{x_1, \ldots, x_m\}$, and the batch normalized data are set to $\boldsymbol{y} = \{y_1, \ldots, y_m\}$. Then the batch normalized algorithm is expressed as [67]

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i, \tag{7}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2, \tag{8}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \tag{9}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta, \tag{10}$$

where $\gamma$ and $\beta$ in the algorithm are the parameters learned in the gradient update. The purpose of this step is to make the result of batch normalization be the same as the original input data, which maintains a possibility to retain the original structure. The parameter $\epsilon$ in Eq. (9) is used to prevent invalid calculation when the variance $\sigma_{\mathcal{B}}^2$ is zero.

The batch normalization technique makes the mean and variance of the input data distribution of each layer in the CNN within a certain range. Thus, each layer of the network does not need to adapt to the change in the distribution of input data, which is conducive to accelerating the learning speed of the model and speeding up the simulation [26]. At the same time, batch normalization suppresses the problem that small changes in parameters are amplified with the deepening of the network layers, making the network more adaptable to model parameters and more stable gradient updates. Besides, due to the use of dropout technique, the number of effective neurons in the model decreases, and the fitting speed of network slows down [67]. Considering that batch normalization has a significant improvement effect on the fitting speed of the model, dropout and batch normalization techniques are often introduced into the structure of the neural network at the same time [68].

The $1 \times 1$ convolution, also known as "network in network," was proposed by Lin *et al.* [69] in 2014. The $1 \times 1$ convolution is a convolution kernel of size $1 \times 1$. In one-dimensional convolution, it is a convolution kernel in which the window length is 1. The $1 \times 1$ convolution is similar to the traditional convolutional layer. After receiving the input data, the operation is gradually performed to obtain the output feature map representation. Its setting parameters commonly used are the number of input channels and the number of output channels [63]. The number of input channels is the feature dimension of the original data. The number of output channels is the same as the number of convolution kernels, which is the feature dimension of the output data.

The $1 \times 1$ convolution recombines multidimensional features of the original data without changing the length of them. It enhances the model's ability to express data features [69]. In addition, if the number of input channels is fewer than the number of output channels, then the $1 \times 1$

TABLE III. The extended techniques used for each model.

| Model | Dropout | Batch norm. | $1 \times 1$ conv. |
|---|---|---|---|
| ConvNet1 | | | |
| ConvNet2 | √ | | |
| ConvNet3 | | √ | |
| ConvNet4 | | | √ |
| ConvNet5 | √ | √ | |
| ConvNet6 | √ | √ | √ |

convolution can be seen as an ascending dimension operation of the original data to increase the feature dimension. This is similar to the effect of data passing through a fully connected layer. Through $1 \times 1$ convolution, the interaction and reorganization of multidimensional features of the original data are realized, and the model's ability to represent data features is enhanced [26].

## V. SIMULATION RESULTS

Based on the enhanced techniques, we extend the basic model of CNN described in Sec. IV A. Dropout, batch normalization, and the $1 \times 1$ convolution are successively added to the basic model. We test these models with data sets 1.1 and 1.2. In the data set 1.1, the SNR range of GW signals is $\rho \in [7, 7.5]$, which makes the signal hard to be detected, representing the edge cases for detection on low SNR signals. The SNR range of data set 1.2 is $\rho \in [7, 15]$, reflecting the detection capability of the model with relatively loud events. The extended models are shown in Table III. Considering that dropout and batch normalization have complementary performance in the overfitting problem, we combine these two techniques in our multi-technique models.

The training set and the testing set of the model are based on data sets 1.1 and 1.2 of training data sets described in Sec. III B. Model parameters and learning strategies are consistent with Sec. IV. We divide 500 samples from the training set as the validation set, which is used to verify the model effect during the training process. When the validation set is input into the model, it does not participate in the gradient update, but is only used to calculate the accuracy and loss function value.

The validation results of each model in the training process are shown in Fig. 7. As shown in the figure, based on data set 1.1 (low SNR), the accuracy of each extended network model is significantly improved compared with that of the basic model (ConvNet1). The validation results

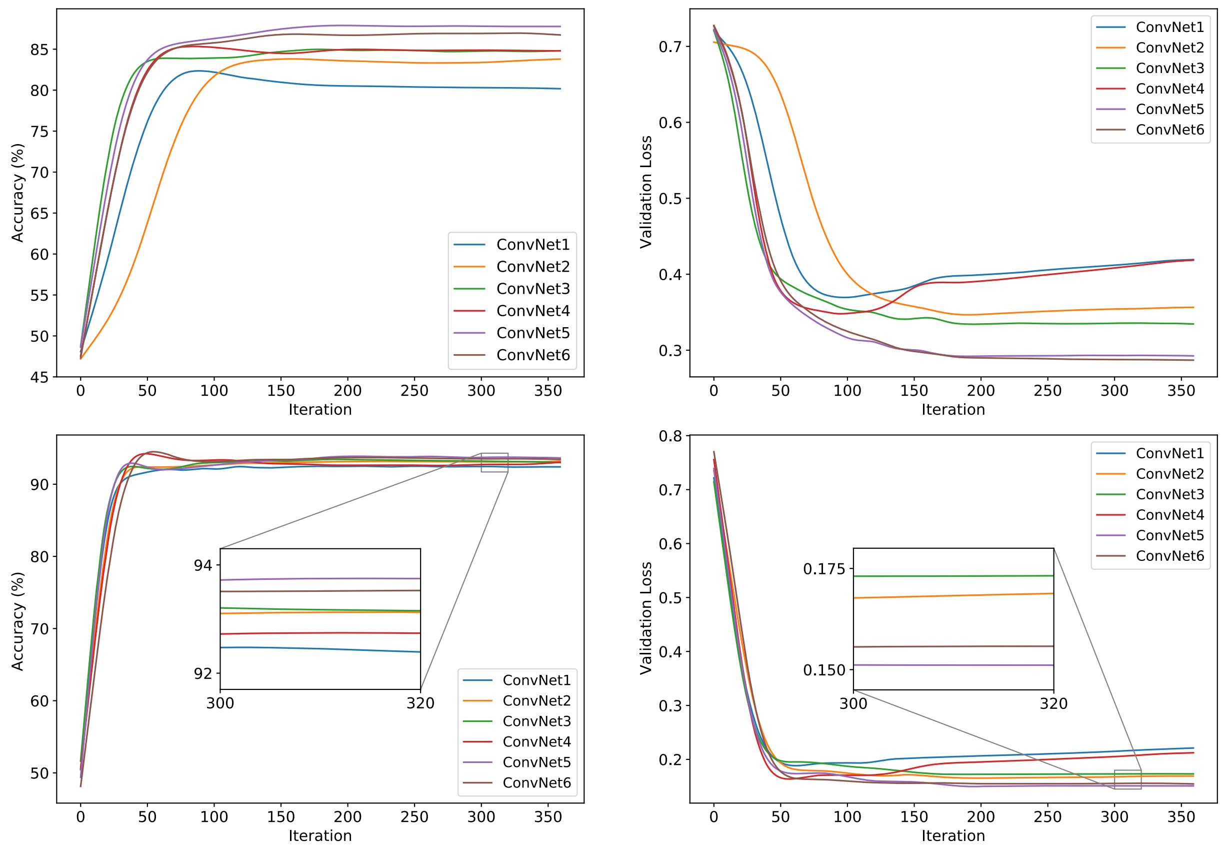


FIG. 7. Model comparison on data sets 1.1 (upper panels) and 1.2 (lower panels). ConvNet1 to ConvNet6 are the models shown in Table III. We use the accuracy (fraction of samples correctly classified) and the validation loss (loss function value in the validation set) as our metrics to track the model performance in the training process.

show that the final accuracy of each network is stable at 83%–87%, while the accuracy of the basic model is below 80%. Among them, ConvNet5, which uses dropout and batch normalization, achieves the highest accuracy in the stable stage after enough iterations. ConvNet6, which uses all improving techniques—namely dropout, batch normalization, and $1 \times 1$ convolution—has an accuracy slightly lower than ConvNet5. ConvNet4, which only uses the $1 \times 1$ convolution, achieves a high accuracy similar to ConvNet6 in the early stage of training, but decreases later on.

Observing the change in the value of the loss function of the validation set, we find that, in the stable stage, ConvNet6 reaches the lowest value in loss, and the loss of ConvNet5 is slightly higher than ConvNet6. Similarly, the minimum values of the loss function of extended models are all smaller than that of the basic model (ConvNet1). Specifically, after applying the dropout to the model, the decline of the loss function values of ConvNet2 is significantly later than the basic model, and in the stable stage, the validation loss of ConvNet2 is lower than that of the basic model. This shows that the dropout technique indeed slows down the fitting speed of network and reduces the overfitting problem. The loss function value of ConvNet3 with batch normalization decreases faster. This means that the fitting speed of the model is accelerated. At the same time, its loss in validation in the late training stage is relatively stable. ConvNet4 with the $1 \times 1$ convolution has a faster fitting speed among the extended models with a single technique, but its late-stage model performance is unstable. In addition, it has the overfitting problem similar to the basic model. ConvNet5 with dropout and batch normalization, and ConvNet6 which uses all three techniques, perform similarly. They both achieve the optimal results in the stable stage. ConvNet6 performs slightly better than ConvNet5 in the stable stage of training.

In data set 1.2, the SNR of the GW signal is larger than that in data set 1.1, which makes the signal detection much easier and reduces the difference between models. In this situation, the extended model is still superior to the basic model. In terms of accuracy, ConvNet5 reaches the optimal value in the stable stage of the late training epoch. Its performance is slightly better than that of ConvNet6. In terms of the value of loss function, ConvNet5 and ConvNet6 have similar results in the stable stage. Actually, ConvNet5 is slightly better than ConvNet6 in the stable stage. In addition, as the SNR distribution range of the data set expands, the differences between the samples increase so that the overfitting problem of the model is reduced.

After training, the models are provided to the testing sets for final evaluation. The testing results are shown in Fig. 8.

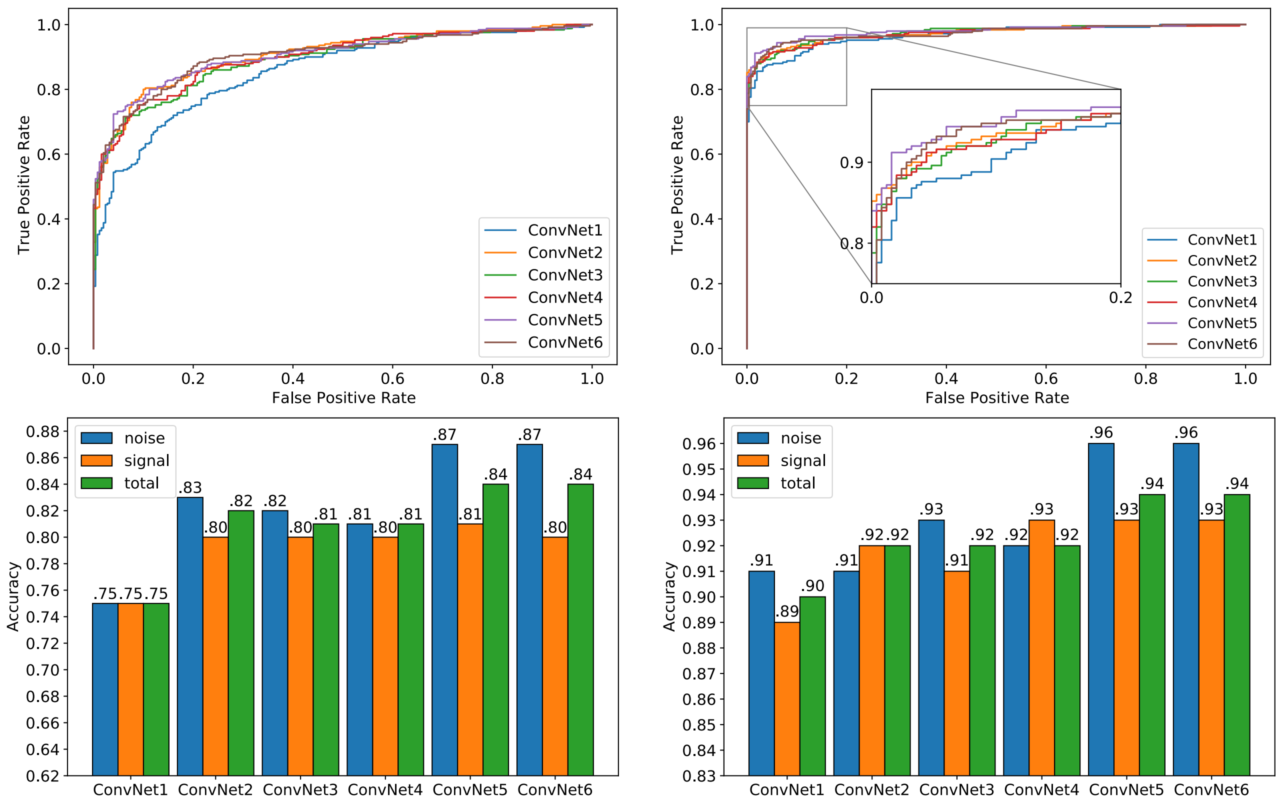


FIG. 8. Results on the testing sets of data sets 1.1 (left panels) and 1.2 (right panels). We use both ROC and the accuracy as our evaluation metrics. As seen in lower panels, the accuracy in identifying noise and signal, as well as the total accuracy, are illustrated separately.

TABLE IV. AUC scores on data sets 1.1 and 1.2. The highest value in each column is highlighted.

| Data set | 1.1 | 1.2 |
|---|---|---|
| ConvNet1 | 0.86 | 0.96 |
| ConvNet2 | 0.90 | 0.97 |
| ConvNet3 | 0.89 | 0.97 |
| ConvNet4 | 0.90 | 0.97 |
| ConvNet5 | **0.91** | **0.98** |
| ConvNet6 | **0.91** | 0.97 |

ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied [70]; see Sec. IV A. When the threshold changes, the curve reflects the fraction of positive samples correctly identified (true positive rate) versus the fraction of negative samples incorrectly identified (false positive rate) [71]. The area under the curve (annotated as AUC) is equal to the probability that a model ranks a randomly chosen positive sample higher than a negative one [70].

The AUC scores of different models on data sets 1.1 and 1.2 are shown in Table IV. It can be seen that on data set 1.1, ConvNet5 and ConvNet6 both achieve the highest accuracy and AUC score, but the classification rate of GW signals of ConvNet6 is slightly lower than that of ConvNet5 (shown in Fig. 8). On data set 1.2, ConvNet5 achieves both the highest accuracy and AUC score.

Based on the above study, we find that the dropout, batch normalization, and $1 \times 1$ convolution techniques can improve the performance of the basic model of CNNs [27]. Among them, the extended models using multiple techniques achieve better results than using a single technique extension. In this work, ConvNet5 using dropout and batch normalization achieves satisfactory results on both data sets. Therefore, we will use this model in the following analysis.

## VI. GENERALIZATION AND ROBUSTNESS

In the work of George and Huerta [27], who firstly introduced the CNN structure into the field of GW data processing, the generalization ability of deep learning algorithms in the task of GW signal detection was discussed. Subsequent studies have pointed out that this ability is a major advantage of deep learning algorithms over the matched filtering [28,35]. As mentioned in Sec. II B, generalization ability refers to the ability of the detection algorithm to respond to signals that are outside of the distribution of training data [49]. We here are the first to study the generalization ability of the CNN model in the task of GW signal detection, including the generalization characteristics of the CNN model in multiple parameter ranges such as masses and aligned spins.

From the discussion in Sec. V, we use ConvNet5 for our following experiments, and hereinafter we refer it as "the model." Considering that the discussion in Sec. V is based only on the performance of the model on the data set where spins are set to zero, we conduct a further test on the ConvNet5 model on data set 4.1. Data set 4.1 introduces spin parameters on the basis of data set 1.2, which is used to test the overall performance of the model in the GW detection task. After 30 epochs of training on data set 4.1, the model reaches an accuracy of 93%, which proves that the model still has a good ability to detect GW signals when considering the spin parameters. In the following subsections, we will test the generalization ability of the model in masses, spins, and the generalization ability of the nonspinning model with spinning data.

### A. Mass parameters

To simplify the investigation, we first study the generalization ability of the model in the GW detection task that only considers mass parameters. The training data set used in this section is data set 2.1 (see Table I). In this data set, the SNR of the GW signal is 8, the mass sampling range of the BBH is $[30\ M_\odot, 60\ M_\odot]$, and the spins are set to zero. After 30 epochs of training, the accuracy on the testing set is 93%. Subsequently, we use the testing data set 2 to test the model on the subdatasets with the mass sampling range of $[10\ M_\odot, 80\ M_\odot]$. The accuracy is shown in the left panel of Fig. 9.

We find that the model works better on subdatasets within the original training parameter range of $[30\ M_\odot, 60\ M_\odot]$. For most of subdatasets beyond the training parameter range (i.e., outside of the range $[30\ M_\odot, 60\ M_\odot]$), high accuracy is still achieved, indicating that the model does have a certain generalization ability in the mass parameter. It can be seen from the data in the figure that the model is more effective for BBHs of high masses, and the accuracy for low-mass BBHs is lower. Studying the signal pattern corresponding to the mass parameter, we find that this is caused by the fact that the model is easier to identify data fluctuations introduced by short signals. As the mass decreases, the GW signal in band becomes longer, which is relatively closer to the characteristics of random noise. This is harder to identify with our CNN model. We show example waveforms of different masses and spins in Fig. 10. It is clear from the upper panels that GW signals of smaller masses are longer; thus, they are harder to be detected in our CNN models. We leave the discussion of further improvement of our CNN models in this parameter space to future studies.

### B. Spin parameters

Now we discuss the generalization ability of the model in spin parameters. We use data set 3.1 as an illustration. The SNR of this data set is 8, the masses of the BBH are $30\ M_\odot$ and $45\ M_\odot$, and the range of the spin parameter $\chi$ is $\chi_i \in [-0.5, 0.5]$ $(i = 1, 2)$. After 30 epochs of training with the

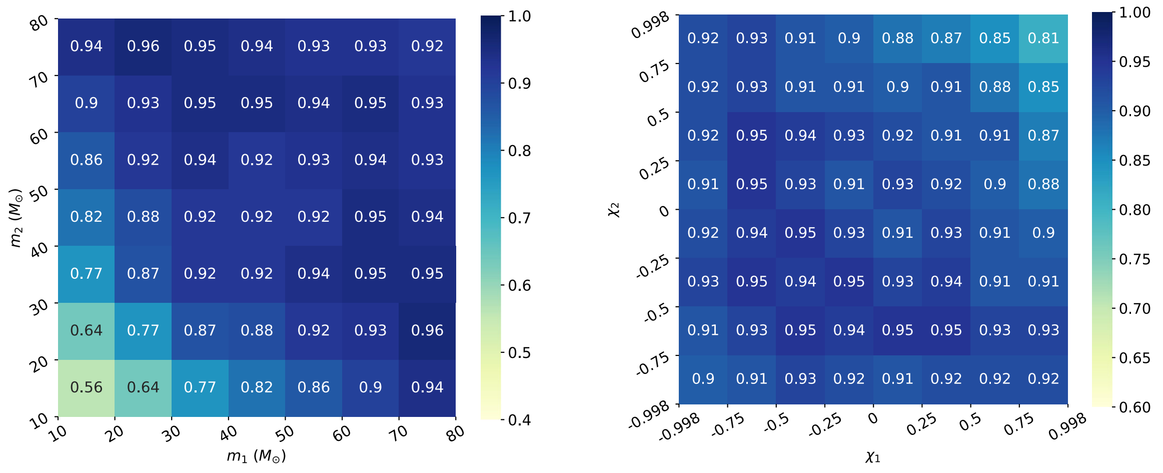


FIG. 9. Model robustness on BBH masses (left) and spins (right). The color bar and numbers are the accuracy of the model tested on different subdatasets (in specific mass/spin ranges) of testing data sets. The mirror part of $m_2 \geq m_1$ is also plotted.

model, the accuracy on the testing set is 92%. Similarly, we use testing data set 3 (see Table II) to test the model on each subdataset with the spin parameter $\chi$ ranging in $[-0.998, 0.998]$.

The accuracy is shown in the right panel of Fig. 9. Similar to the generalization ability in the mass parameter, the model performs well on the subdatasets in the original parameter range. It also has good accuracy on the subdatasets beyond the parameter range. Among them, the model performs slightly worse on the subdatasets with larger positive spins. This is also consistent with the case in Sec. VI A. As shown in the lower panels of Fig. 10, the spin parameters of the BBHs are taken as (0.998, 0.998), (0, 0), $(-0.998, -0.998)$, respectively, from left to right. It is clear that with the increase of the $\chi$ value, the GW signals become longer (the "hang-up" effect), thus harder to be detected with our CNN model. Further improvements in this aspect will be subject to future studies.

### C. Robustness of nonspinning model on spinning data

What if the model is only trained with nonspinning GW signals, and it is tested with spinning signals? We use the data set 1.2 (see Table I) for training that only uses the mass parameters. After 30 epochs of training, we test the model on the testing data set 4 (see Table II) which contains spin parameters as well. We obtain an accuracy of ~92%. The accuracy of the model is only slightly reduced compared to the accuracy of ~94% in the data set 1.2 where only the mass parameter is included. It shows the generalization ability of the model that is only trained with nonspinning GWs, but tested in data containing spin parameters.

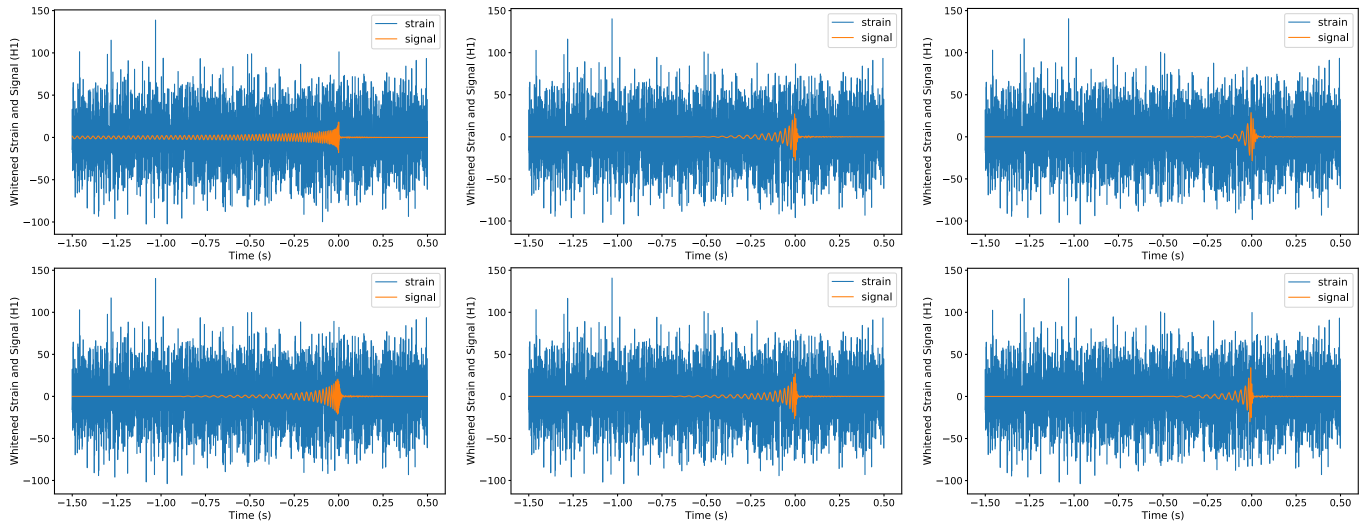


FIG. 10. Simulated GW strain examples of different masses and spins. The upper panels are the strain examples of increasing masses from left to right: ($10\ M_\odot, 10\ M_\odot$), ($45\ M_\odot, 45\ M_\odot$), and ($80\ M_\odot, 80\ M_\odot$). The lower panels contain examples of different spins (from left to right): (0.998,0.998), (0,0), and $(-0.998, -0.998)$.

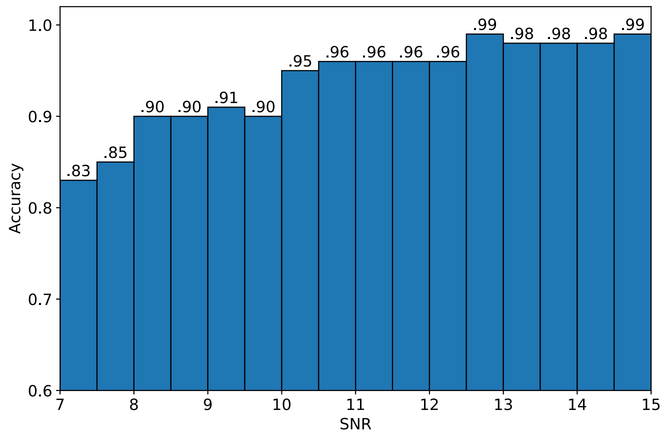


FIG. 11. Robustness of CNN models that are only trained with nonspinning GWs on the spinning GW data for different SNRs.

We further use the testing data set 5 (see Table II) to test the model on subdatasets with different SNR distribution. The result is shown in Fig. 11. As shown in the figure, the model has the best detection efficiency on data with high SNR and it is slightly worse on data with low SNR. However, on the subdataset of testing data with SNR in the range of [7.0, 7.5], the model still maintains an accuracy of ∼83%. This shows that the CNN model which is trained only with nonspinning data is robust to data containing nonzero spin parameters.

### D. Conclusion

From the above study, it can be seen that the CNN model shows good generalization ability for data of different parameter ranges, which is a major advantage over the matched-filtering method in using the CNN model. The matched-filtering method is based on the existing template bank. Its search and response to GW signals are limited to the existing waveforms. The GW signals beyond the existing waveforms cannot be easily detected. The generalization ability of the CNN model in the task of GW signal detection will help us to discover signals beyond the existing templates. It may also be able to detect signals that imprint orbital eccentricity [72], orbital precession [73], and deviations from general relativity [74]. Such a generalization ability will undoubtedly play an essential role in searches of GW signals beyond what we have in the template bank. More studies are needed along this direction.

## VII. DISCUSSION

For the first time, this work specifically studied the effects on CNN models brought by the improving techniques—such as dropout, batch normalization, and $1 \times 1$ convolution—in the task of GW signal detection. Comparisons of models with the combination of different improving techniques were made. Our simplified experiments show that dropout, batch normalization, and the $1 \times 1$ convolution can significantly improve the accuracy of the basic model [27]. In addition, we conducted specific experiments and discussed the generalization ability of CNNs in GW signal detection tasks. Experiments show that the CNN model has good generalization ability over multiple parameter ranges, including masses and aligned spins. It is a major advantage of the deep learning models compared with the matched-filtering method.

In recent years, with the rapid development in the field of deep learning, deep learning models and algorithms have emerged one after another. CNN is only a basic model in the market of deep learning though, it has shown excellent performance in many fields such as image recognition and signal detection [26]. As the difficulty of the task continues to increase, the requirements for feature extraction and processing are getting higher and higher. The network depth of the CNN is limited by the gradient propagation problem, resulting in a limitation on its performance [26]. The proposal of the residual network structure greatly alleviates the problem of gradient propagation in CNNs and makes the depth limit of CNN models rapidly expand [75]. In the field of time series processing, the "long short term memory" model is widely used in multiple tasks, such as speech processing and stock price prediction, and has achieved excellent results [76]. At present, some studies have introduced the above structure into the process of GW data processing [32,34]. However, in the task of GW signal detection, there is still a lack of unified investigation on these models. Comparing these models (including CNN models) on GW signal detection tasks, such as the recognition rate, model size, running time, and other indicators, is crucial to find an optimal model structure suitable for real-world task. It is a worthwhile direction for future studies.

For the sake of simplifying the investigation, this work only uses simulated white noise to construct GW sample data. In order to make the model more in line with the characteristics of reality detection, real noise construction from aLIGO/Virgo data can be conducted. At the same time, the data used in this paper are only for experimental purposes, so the amount of data is limited. By increasing the number of samples in the data set, the performance of the model can be further improved. Ensemble learning is also an optimization technique that is worth considering, where the basic idea is to train multiple neural network models, and to use voting and other common decision-making strategies in order to make up for the decision-making defects of a single model and to improve the decision-making ability of the overall model [77]. These aspects deserve detailed studies on their own.

## ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grants No. 11975027, No. 11991053, No. 11690023, No. 11721303), the

Young Elite Scientists Sponsorship Program by the China Association for Science and Technology (2018QNRC001), the Max Planck Partner Group Program funded by the Max Planck Society, and the High-Performance Computing Platform of Peking University. It was partially supported by the Strategic Priority Research Program of the Chinese Academy of Sciences through Grant No. XDB23010200.

[1] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. Lett. **116**, 061102 (2016).
[2] B. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), in *Centennial of General Relativity: A Celebration*, edited by C. A. Z. Vasconcellos (World Scientific, Singapore, 2017), pp. 291–311.
[3] J. Liu, G. Wang, Y.-M. Hu, T. Zhang, Z.-R. Luo, Q.-L. Wang, and L. Shao, Chin. Sci. Bull. **61**, 1502 (2016).
[4] B. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. X **9**, 031040 (2019).
[5] A. H. Nitz, C. Capano, A. B. Nielsen, S. Reyes, R. White, D. A. Brown, and B. Krishnan, Astrophys. J. **872**, 195 (2019).
[6] R. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. D **102**, 043015 (2020).
[7] R. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Astrophys. J. Lett. **896**, L44 (2020).
[8] B. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Astrophys. J. Lett. **892**, L3 (2020).
[9] R. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. Lett. **125**, 101102 (2020).
[10] R. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), arXiv:2010.14527.
[11] B. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. Lett. **119**, 161101 (2017).
[12] L. S. Finn, Phys. Rev. D **46**, 5236 (1992).
[13] K. Cannon *et al.*, Astrophys. J. **748**, 136 (2012).
[14] S. A. Usman *et al.*, Classical Quantum Gravity **33**, 215004 (2016).
[15] B. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. D **93**, 122004 (2016); **94**, 069903(A) (2016).
[16] R. Smith, S. E. Field, K. Blackburn, C.-J. Haster, M. Pürrer, V. Raymond, and P. Schmidt, Phys. Rev. D **94**, 044031 (2016).
[17] I. Harry, S. Privitera, A. Bohé, and A. Buonanno, Phys. Rev. D **94**, 024012 (2016).
[18] N. J. Cornish and T. B. Littenberg, Classical Quantum Gravity **32**, 135012 (2015).
[19] A. J. Chua, C. R. Galley, and M. Vallisneri, Phys. Rev. Lett. **122**, 211101 (2019).
[20] A. J. Chua and M. Vallisneri, Phys. Rev. Lett. **124**, 041102 (2020).
[21] N. Mukund, S. Abraham, S. Kandhasamy, S. Mitra, and N. S. Philip, Phys. Rev. D **95**, 104059 (2017).
[22] R. E. Colgan, K. R. Corley, Y. Lau, I. Bartos, J. N. Wright, Z. Márka, and S. Márka, Phys. Rev. D **101**, 102003 (2020).
[23] E. Cuoco, J. Powell, M. Cavaglià *et al.*, Mach. Learn. Sci. Technol. **2**, 011002 (2020).
[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Commun. ACM **60**, 84 (2017).
[25] J. Schmidhuber, Neural Netw. **61**, 85 (2015).
[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016), http://www.deeplearningbook.org.
[27] D. George and E. Huerta, Phys. Rev. D **97**, 044039 (2018).
[28] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).
[29] P. G. Krastev, Phys. Lett. B **803**, 135330 (2020).
[30] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, arXiv:1909.06296.
[31] H. Shen, E. Huerta, Z. Zhao, E. Jennings, and H. Sharma, arXiv:1903.01998.
[32] H. Shen, D. George, E. A. Huerta, and Z. Zhao, in *ICASSP 2019: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, Brighton, United Kingdom, 2019), pp. 3237–3241.
[33] C. Chatterjee, L. Wen, K. Vinsen, M. Kovalam, and A. Datta, Phys. Rev. D **100**, 103025 (2019).
[34] C. Dreissigacker, R. Sharma, C. Messenger, R. Zhao, and R. Prix, Phys. Rev. D **100**, 044009 (2019).
[35] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Phys. Rev. D **100**, 063015 (2019).
[36] D. George, H. Shen, and E. Huerta, Phys. Rev. D **97**, 101501 (2018).
[37] M. Chen, Y. Zhong, Y. Feng, D. Li, and J. Li, Sci. China Phys. Mech. Astron. **63**, 129511 (2020).
[38] J. P. Marulanda, C. Santa, and A. E. Romano, Phys. Lett. B **810**, 135790 (2020).
[39] H. Wang, S. Wu, Z. Cao, X. Liu, and J.-Y. Zhu, Phys. Rev. D **101**, 104003 (2020).
[40] T. M. Mitchell, *Machine Learning* (McGraw-Hill, New York, 1997).
[41] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Springer-Verlag, Berlin, Heidelberg, 2006).
[42] T. Kohonen, Neural Netw. **1**, 3 (1988).
[43] W. S. McCulloch and W. Pitts, Bull. Math. Biophys. **5**, 115 (1943).
[44] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, arXiv:1811.03378.
[45] V. Nair and G. E. Hinton, in *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Omnipress, Madison, WI, 2010), pp. 807–814.
[46] J. Hopfield, Proc. Natl. Acad. Sci. U.S.A. **79**, 2554 (1982).

[47] S. Sra, S. Nowozin, and S. Wright, *Optimization for Machine Learning*, Neural Information Processing Series (MIT Press, Cambridge, MA, 2012).

[48] S. Urolagin, K. V. Prema, and N. V. S. Reddy, in *Proceedings of the 2011 International Conference on Advanced Computing, Networking and Security, ADCONS'11* (Springer-Verlag, Berlin, Heidelberg, 2011), pp. 171–178.

[49] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning, Second Edition*, Adaptive Computation and Machine Learning Series (MIT Press, Cambridge, MA, 2018).

[50] S. Lawrence and C. L. Giles, in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, Vol. 1* (IEEE, New York, 2000), pp. 114–119.

[51] D. M. Hawkins, J. Chem. Inf. Comput. Sci. **44**, 1 (2004).

[52] K. Yamaguchi, K. Sakamoto, T. Akabane, and Y. Fujimoto, in *The First International Conference on Spoken Language Processing, ICSLP 1990, Kobe, Japan, November 18-22, 1990* (ISCA, Kobe, Japan, 1990).

[53] D. Ciregan, U. Meier, and J. Schmidhuber, in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Providence, RI, USA, 2012), pp. 3642–3649.

[54] H. H. Aghdam and E. J. Heravi, *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*, 1st ed. (Springer Publishing Company, Incorporated, New York, 2017).

[55] C. Cutler and É. E. Flanagan, Phys. Rev. D **49**, 2658 (1994).

[56] A. Bohé, L. Shao, A. Taracchini, A. Buonanno *et al.*, Phys. Rev. D **95**, 044028 (2017).

[57] https://pycbc.org/

[58] A. Nitz, I. Harry, D. Brown, C. M. Biwer, J. Willis *et al.*, gwastro/pycbc: Pycbc release v1.16.9 (2020).

[59] https://github.com/lscsoft/lalsuite

[60] D. Shoemaker, LIGO Document T0900288-v3 (2010).

[61] C. E. Shannon Bell Syst. Tech. J. **27**, 379 (1948).

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

[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury *et al.*, arXiv:1912.01703.

[64] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, arXiv:1410.0759.

[65] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, arXiv:1207.0580.

[66] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, J. Mach. Learn. Res. **15**, 1929 (2014), https://dl.acm.org/doi/10.5555/2627435.2670313.

[67] S. Ioffe and C. Szegedy, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning—Volume 37, ICML15* (ICML, Lille, France, 2015), pp. 448–456.

[68] X. Li, S. Chen, X. Hu, and J. Yang, in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2019), pp. 2677–2685.

[69] M. Lin, Q. Chen, and S. Yan, arXiv:1312.4400.

[70] T. Fawcett, Pattern Recogn. Lett. **27**, 861 (2006).

[71] D. M. W. Powers, J. Mach. Learn. Technol. **2**, 37 (2011).

[72] X. Liu, Z. Cao, and L. Shao, Phys. Rev. D **101**, 044049 (2020).

[73] S. Babak, A. Taracchini, and A. Buonanno, Phys. Rev. D **95**, 024010 (2017).

[74] L. Shao, N. Sennett, A. Buonanno, M. Kramer, and N. Wex, Phys. Rev. X **7**, 041025 (2017).

[75] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Las Vegas, NV, USA, 2016), pp. 770–778.

[76] S. Hochreiter and J. Schmidhuber, Neural Comput. **9**, 1735 (1997).

[77] D. Opitz and R. Maclin, J. Artif. Intell. Res. **11**, 169 (1999).