Convolutional neural network for gravitational-wave early alert: Going down in frequency

Grégory Baltus,[1,*] Justin Janquart,[2,3,†] Melissa Lopez,[2,3,‡] Harsh Narola,[2,3,§] and Jean-René Cudell[1,‖]

[1]*STAR Institute, Bâtiment B5, Université de Liège, Sart Tilman B4000 Liège, Belgium*
[2]*Nikhef, Science Park 105, 1098 XG Amsterdam, Netherlands*
[3]*Institute for Gravitational and Subatomic Physics (GRASP), Utrecht University, Princetonplein 1, 3584 CC Utrecht, Netherlands*

(Received 11 May 2022; accepted 13 July 2022; published 2 August 2022)

We present here the latest development of a machine-learning pipeline for premerger alerts from gravitational waves coming from binary neutron stars (BNSs). This work starts from the convolutional neural networks introduced in [Baltus *et al.*, Phys. Rev. D **103**, 102003 (2021)] that searched for the early inspirals in simulated Gaussian noise colored with the design-sensitivity power-spectral density of LIGO. Our new network is able to search for any BNS with a chirp mass between 1 and 3 $M_\odot$, it can take into account all the detectors available, and it can see the events even earlier than the previous one. We study the performance of our method in three different scenarios: colored Gaussian noise based on the O3 sensitivity, real O3 noise, colored Gaussian noise based on the predicted O4 sensitivity. We show that our network performs almost as well in non-Gaussian noise as in Gaussian noise: our method is robust with respect to glitches and artifacts present in real noise. Although it would not have been able to trigger on the BNSs detected during O3 because their signal-to-noise ratio was too weak, we expect our network to find around 3 BNSs during O4 with a time before the merger between 3 and 88 s in advance.

DOI: 10.1103/PhysRevD.106.042002

I. INTRODUCTION

Multimessenger astrophysics (MMA) makes use of messengers from different forces of the Universe to provide a wealth of information about various astrophysical processes. From previous investigations it is well known that the combination of at least two of these signals gives qualitatively different and complementary types of information, capable of probing down to the densest and most energetic regions of cosmic objects, which were hidden from astronomers' sight up until now [1–3].

In the context of gravitational waves (GW) combined with other astrophysical signals, it has long been suggested that short gamma ray burst (GRB) might be related to binary neutron star mergers [4], which fall in the sensitivity band of second generation ground based-detectors [5–7]. Several studies investigated the expectations of electromagnetic (EM) follow-up efforts during the Advanced LIGO and Virgo era of compact binary coalescence (CBC) [8,9]. On August 17, 2017 the Fermi γ-ray Burst Monitor [10] announced the detection of a GRB, later designated as GRB170817A [11].

Approximately 6 min later, a GW candidate, later relabeled as GW170817, was registered in low latency based on a single-detector analysis of the Advanced Laser Interferometer Gravitational-wave Observatory (LIGO) Hanford data and disseminated through a γ-ray Coordinates Network (GCN) Notice. A rapid re-analysis of data from Hanford, Livingston, and Virgo confirmed a highly significant, coincident signal [12]. An extensive observing campaign was launched across the electromagnetic spectrum in response to the Fermi-GBM and LIGO–Virgo triggers, which led to the detection of the kilonova associated with GW170817, later called AT 2017gfo [13].

In recent times, there has been a sparkling interest in early warning (or premerger) alerts of BNS in the field of GW for EM and astroparticle follow-ups [14–24], since the radiation emitted from these systems enters the sensitive region of the interferometers during the inspiral phase [25,26]. Assuming all BNS's produce a short GRB with an x-ray, optical and radio afterglow, an LVK network, and a top-hat jet model, Ref. [27] predicts rates for the joint detections of 0.02–27 per year for X-ray band, 0.01–19 per year for optical band, and 0.02–25 per year for radio band, respectively, at design sensitivity for a three detector network [27]. It is relevant to note that the large uncertainty is due to the fact that BNS merger rate is

*gbaltus@uliege.be
†j.janquart@uu.nl
‡m.lopez@uu.nl
§h.b.narola@uu.nl
‖jr.cudell@uliege.be

not well constrained. The improving sensitivity of second-generation detectors and the even better sensitivity for the third-generation detectors, such as Cosmic Explorer and the Einstein Telescope [28,29] will lead, via an increase in the signal-to-noise ratio (SNR), to a major improvement in the early-detection and sky-localization capabilities [21,22,30]. Another key element to develop MMA further is the design of low-latency pipelines for the production of real-time GW alerts, or even premerger alerts. The current state-of-the-art employs matched filtering techniques [8] to perform online analyses via the pipelines GstLAL [31–34], PyCBC [19,35–40], MBTAOnline [41,42], cWB [43–45], and SPIIR [46]. We refer the reader to [47] and [48–50] for a summary of the low-latency efforts carried out by the LIGO-Virgo collaboration during the second and third observing runs. Recent investigations in the GW field have focused on Machine Learning (ML) algorithms, due to their success in different tasks and domains. The main advantage of ML techniques is their rapidity because most of the computations are made during the training stage. A widely used ML method for pattern recognition is based on convolutional neural networks (CNNs) [51], in the context of GW it has been applied to different tasks such as CBC identification [52–56], burst detection [57–60], sky localization [61–63], glitch classification [64,65] and synthetic data generation [66,67]. See [68] for a review on this topic.

ML methods have also emerged as a new tool in the context of early warning [14,16], allowing us to flag prompt triggers for GW candidates. The final goal of this work is to detect BNS signals before the merger. To do that, we have design a single CNN that takes as input the time-series data from all the online detectors and returns a classification between two classes: pure noise or noise plus inspiral. In this paper, we build on our previous work [14], improving on the techniques previously developed, and testing them on more realistic scenarios: we use real O3 noise, as well as the data from all available detectors. In addition, we retrained our network on predicted O4 noise and give expected efficiencies for this run.

The details of the differences with [14] are as follows:

(i) the addition of the spin effect to the BNS waveforms;
(ii) a uniform sky location of the injections;
(iii) the injection of simulated BNS signals in simulated O3 noise, real O3 noise, and simulated O4 noise;
(iv) a decrease of the minimal cutoff frequency from 20 Hz to 10–15 Hz;
(v) a fixed input-signal duration of 300 s with a sampling frequency of 512 Hz that allows us to analyze any BNS signal for all allowed neutron star masses;
(vi) the implementation of curriculum learning [69].

This paper is organized as follows: in Sec. II, the method is explained. Section II A introduces the definition of the SNR and the partial inspiral signal-to-noise ratio (PISNR) used in this work, as well as the relation between the frequency of a waveform and the time before the merger. The description of the data generation and the training strategy is made in Sec. II B. The last part of this section, II C, describes the architecture of the CNN used in this paper. Section III presents the results and the performance of our method in the three types of noise, as well as studying the number of BNS that are expected to be found in advance by our network in O4. Finally, we give our conclusions in Sec. IV.

II. METHOD

A. Loudness and frequency evolution of the signal

In GW-searches, the matched-filtering SNR (ρ) [19,70] is used to verify how well a template matches the data. The SNR definition follows that of the FINDCHIRP algorithm [35] as implemented in PyCBC [71]. One first transforms the signal $s(t)$ and templates $h(t)$ to frequency space:

$$\tilde{h}(f) = \int_{-\infty}^{\infty} h(t) e^{-2\pi i f t} dt \tag{1}$$

and similarly for $\tilde{s}(f)$. One can define the matched filtering output [70]

$$x(t) = 4\mathcal{R} \int_0^{\infty} \frac{\tilde{s}(f)^* \tilde{h}(f)}{S_n(f)} e^{2\pi i f t} df \tag{2}$$

where $S_n(f)$ is the one-sided noise strain power spectral density (PSD) of the detector and the * superscript denotes complex conjugation. One can show that this matched filtering output still depends on the phase at a reference time of the signal, for instance at the time when it enters the frequency band of the interferometer. It is possible to minimize $x(t)$ analytically with respect to that phase [35,72] and express the result via the complex $z(t)$:

$$z(t) = 4 \int_0^{\infty} \frac{\tilde{s}(f)^* \tilde{h}(f)}{S_n(f)} e^{2\pi i f t} df \tag{3}$$

as

$$\min(x(t)) = |z(t)|$$

The variance of $|z(t)|$ is given by:

$$\sigma^2 = 4 \int_0^{\infty} \frac{\tilde{h}(f)^* \tilde{h}(f)}{S_n(f)} df. \tag{4}$$

The signal-to-noise ratio is then taken to be [72]

$$\rho(t) = \frac{|z(t)|}{\sigma}. \tag{5}$$

For a network of N detectors, identified by an index $i = 1...N$, one defines the network SNR as

$$\rho_{\mathrm{net}}(t) = \sqrt{\sum_{1}^{N} \rho_i^2(t)}. \tag{6}$$

The SNR is a key quantity for the searches based on matched filtering, since it describes the amount of overlap between a template and an unknown signal. In these searches, the strategy is to create a template bank of precomputed waveforms and use it to calculate the SNR over all the data strain. As a first step, a trigger is created when the SNR reaches a maximum value higher than a given threshold. After that, it undergoes a statistical treatment to be confirmed as a GW candidate [19]. As we are interested in searching for the early inspiral, a more meaningful indicator will be the partial-inspiral signal-to-noise ratio. It is defined as the SNR in which the template h is the partial template that contains only the fraction of the inspiral part of the waveform that our network tries to identify. For more details about the PISNR, and how it evolves depending on the length of the template, we refer to Sec. II. A. of Ref. [14].

At the lowest order in velocity, the frequency f at a time t depends on the chirp mass $\mathcal{M}_c$ of the system and the merger time t_m:

$$f(t) = \frac{1}{\pi}\left(\frac{G\mathcal{M}_c}{c^3}\right)^{-5/8}\left(\frac{5}{251}\frac{1}{(t_m - t)}\right)^{3/8}. \tag{7}$$

So for a given chirp mass, if we say that we can detect an event $(t_m - t)$ s before the merger, it is equivalent to say that we detect the signal when the maximum frequency is $f(t)$. Figure 1 represents the time and frequency evolution for a GW.

B. Data and training strategies

In our previous work [14], we have shown the possibility to detect the early inspiral of a BNS injected in Gaussian noise. In this work, we want to turn to a more realistic scenario, using real O3 noise. To investigate the difference in performance between Gaussian and real noise, we also inject the signals in colored Gaussian noise generated from the O3 representative PSD. In addition, to assess the performance of our network in future observation runs, we consider colored Gaussian noise generated from the predicted O4 PSD.

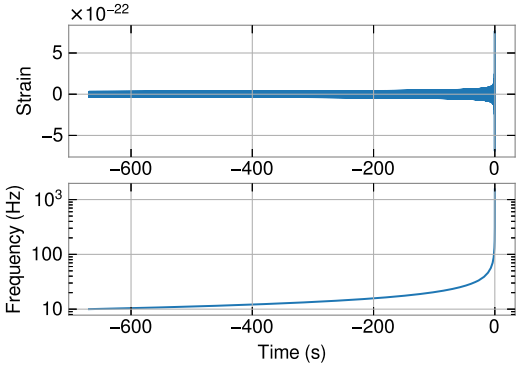

FIG. 1. The top figure represents a GW signal corresponding to two objects of mass 1.8 $M_\odot$. The bottom figure represents the evolution in frequency for this binary.

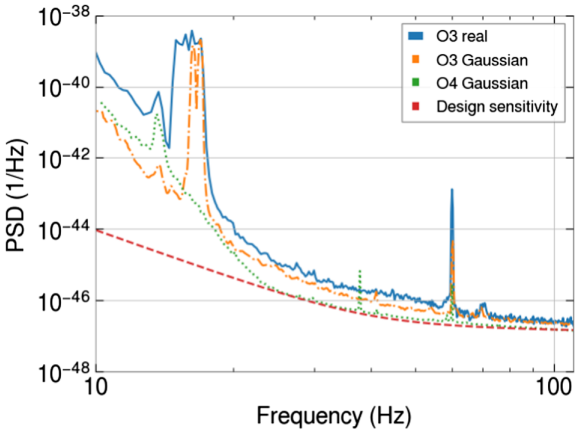

FIG. 2. Representation of the different PSDs, for the Livingston detector, used to generate the different datasets. We also show the design sensitivity PSD provided by PyCBC [71] used in [14].

The corresponding PSDs are represented in Fig. 2. To generate a frame of simulated O3 Gaussian noise, we use the PSDs from [73], provided by PyCBC [71].[1] To obtain data of O3, we directly download the strain of the detectors [48–50] using the *GWpy* package [74]. To be closer to a real time search, these downloaded strains are the ones recorded in low-latency, meaning that they are not filtered and cleaned as extensively as the final noise.[2] To generate the O4 Gaussian noise, we use the predicted O4 PSD coming from the observing scenarios [73,75].[3]

Since the problem at hand can be solved as a classification task, we need a dataset containing two classes: noise and noise plus inspiral, also known as injections. For the injections, we generate waveforms using the approximant *SpinTaylorT4* [76]. We choose the component masses to be uniformly distributed between 1 and 3 solar masses to cover all the possible BNS systems [77]. The sources are uniformly distributed over the sky, and we also

[1]The PSD used for Gaussian O3 LIGO is *aLIGOaLIGO140MpcT1800545*, the one for Virgo is *aLIGOAdVO3LowT1800545*, both are provided by LIGO and Virgo and implemented via PyCBC [71].

[2]To download the real O3 data, we use the channels H1:GDS-CALIB_STRAIN, L1:GDS-CALIB_STRAIN, V1:Hrec_hoft_16384 Hz, and the frame type: H1_llhoft, L1_llhoft, V1Online in *GWpy*.

[3]The LIGO and Virgo PSDs used for O4 correspond to the ones shown in Fig. 1 of [73], with the BNS detector horizon at 160 Mpc for the LIGO detectors and the horizon at 120 Mpc for the Virgo detector.

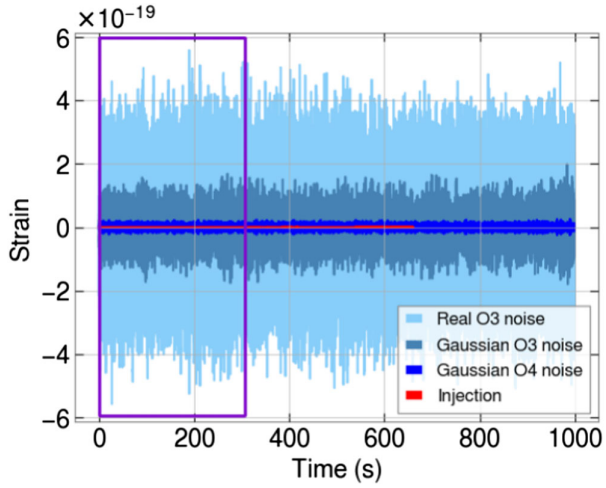

FIG. 3. Representation of the different types of noises for the Handford detector used together with an injection similar to GW170817, i.e., with neutron star masses of 1.46 and 1.27 $M_\odot$ [78]. Note that only the part in the rectangle is passed to the network.

include the spin effects. With these parameters and a lower frequency of 10 Hz, the simulated signal is always longer than 300 s. In such a way, the inputs of the network contain only the early inspiral part (see Fig. 3 for an illustration). After injecting the simulated signal into the noise the frames are whitened, and we apply a low-pass filter at 100 Hz and a high-pass filter at 10 Hz. For O3 real noise, some significant peaks can appear in the whitened strain due to non-Gaussian effects. In our approach, these effects are vetoed by zeroing them out, see the Appendix. Afterwards, the final frame is renormalized, making all the values in the frame between −1 and 1. This will be the input data of the network, and we refer to a single sample as a *frame*.

For the training and testing, we choose a distribution in distance such that the distribution in PISNR is an inverse Gaussian with a mean of thirty-five and a scale of one hundred.[4] Despite having a large dataset containing one million frames, we have observed a low performance when we decrease the maximum frequency to ∼25 Hz. This is because the CNN is good to detect a variation of frequency, and for earlier inspiral phase, the signal becomes more monochromatic.

To be able to detect events earlier, it is key to decrease the maximum frequency seen by our model. For this aim, we change the training strategy and use curriculum learning [69] as a function of the maximal frequency seen by the network, as it has shown an increase of the performance as a function of the SNR in previous works [59]. The principle of curriculum learning is to train the network first on easier data (on data with a high maximum frequency), and then gradually increasing the difficulty (on data with a lower maximum frequency). The network is then iteratively trained on each training set. To prevent the network from forgetting what it has learned, we keep all the data of the previous steps while adding the new ones. To that effect, we generate five different training sets. The parameter distributions for the injections stay the same, except for the maximal frequency seen by the network. This parameter is now chosen as a Gaussian distribution with a standard deviation of 2.5 Hz, and different mean depending on the datasets. More information about these datasets can be found in Table I. Each training set contains 20000 frames and half of them contain an injection. Note that 20% of each training set was used for validation during the training. For each step, we train for six epochs as it was enough to make the loss converge without facing overfitting. The use of curriculum learning allows to improve the performance on dataset 3, 4, 5 with maximum frequency of, respectively, 30, 25, 20 Hz, while maintaining the performance at higher frequencies.

The training on the real noise data was done in a similar way. Note that we have done the training with noise coming only from O3a, meaning the first half of O3 [48]. We have vetoed the time of the real events from the GWTC-2.1 catalog [48] not to train on them, as most of them were BBH. For all the testing, we used noise coming from O3b, the second half of O3 [49,50]. During O3 there are times when not all detectors are online. To take this fact into account, when a certain detector is offline, we fill the CNN entry corresponding to the detector with a vector of zeros. In this way, our network is able to perform the search regardless of the number of detectors available.

For the training parameters, we use a batch size of 50. For each step of curriculum learning we train for 6 epochs, it was enough to make the loss converge. The learning rate is 8×10^{-5} and the optimizer is ADAMAX with a weight

TABLE I. Each dataset corresponds to a value for the maximum frequency seen by the networks, which in turn leads to a minimum frequency and a time before the merger (TBM). The numbers shown for the maximum and minimum frequency are the mean value in each dataset. The maximum and minimum TBMs are the TBM for two objects of respectively 3 $M_\odot$ and 1 $M_\odot$.

Dataset	Max frequency (Hz)	Min frequency (Hz)	Min TBM (s)	Max TBM (s)
Dataset 1	40	12.9	7	44
Dataset 2	35	12.8	10	63
Dataset 3	30	12.6	15	95
Dataset 4	25	12.3	24	115
Dataset 5	20	11.7	45	280

[4]We found that the inverse Gaussian (Wald) distribution fits better our goal. Indeed, this distribution gives a few very high PISNR events that enable the network to start its learning process.

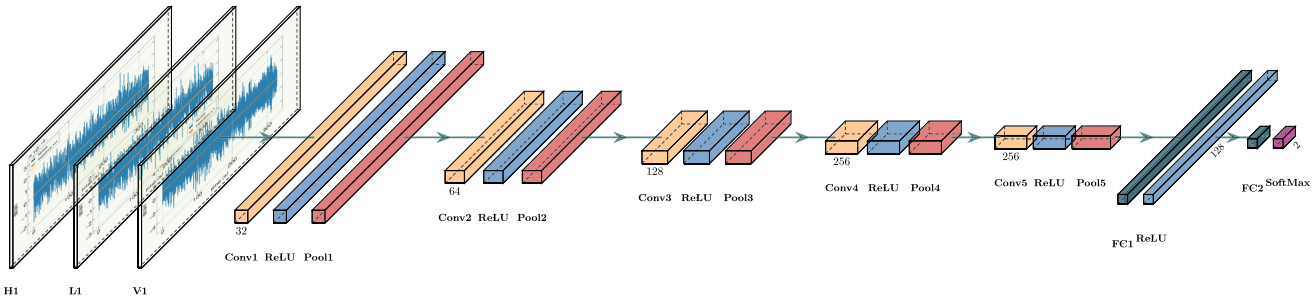

FIG. 4. Representation of the CNN architecture, the yellow layers are the convolutions, the blue ones are the ReLU activation, the red ones are the pool layers, the purple ones are the dense layers, and the dark purple is the final softmax layer. The number under each layer represents the number of channels.

decay of 10^{-5}. ADAMAX is a variant of ADAM, based on the infinity norm [79]. In our previous work [14], we have seen that using ADAMAX leads to a faster the convergence of the training.

We use the weighted cross-entropy loss [80]. At first, we employed the cross-entropy loss, which is standard for classification problems. However, this led to a large number of false positives. To remedy that we decided to weigh this loss [59] by a factor 0.4 for the frames with an injection. This reduces the chances that the network classifies a frame containing only noise as an event, so it reduces the number of false positives. We tried multiple values for the weight and found that for the task at hand a factor of 0.4 translates into a reduction of the number of false positive while maintaining the number of true positive. The duration of the training is about one day on a NVIDIA Tesla V100-PCIE-16Gb GPU.

C. Description of the network

The architecture of the network is similar to the one in Ref. [14], where multiple trial and errors were made to end up with the architecture. A representation of the neural network is given in Fig. 4, we use the PyTorch package to create the architecture [81]. The network takes 300 s of data for each available detector. In other words, it has three input channels, each corresponding to one of the three detectors (Hanford, Livingston, and Virgo).[5] It is composed of a batch normalization layer, followed by 5 blocks composed of a convolution layer, a ReLU activation, and a pool layer. For the convolution, the kernel sizes are successively 16, 8, 4, 8, 16. For the pool layers, the kernel size is always set to 4. The stride is set to 1 for the convolution layers and 4 for the pool layers. After these blocks, we add two linear layers with sizes of respectively 128 and 2 interspersed by a ReLU activation. The final layer is a softmax layer that returns a probability vector.

[5]The Conv1D layer as implemented in PyTorch allow us to give as input any number of channels, see https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html [81].

III. RESULTS

A. Performance of the network

After the training, the testing sets come from the same distribution as the training sets, see Table I. The other parameter distributions are the same as for the training sets. Each of the test sets contains 4400 frames, half of which are pure noise and half noise plus injection. The total size of the test sets for a type of noise is then 22000 frames.

The efficiency of our network for the different steps of curriculum learning can be seen in Fig. 5. We define the true alarm probability (TAP) and the false alarm probability (FAP) as Eq. (7) in our previous work [14]. In Fig. 5, we represent the three lowest maximum frequencies dataset of curriculum learning, as the higher maximum frequencies have performances similar to the 30 Hz dataset. For the datasets with a maximum frequency > 25 Hz, an efficiency of 50% is obtained at $\sim$15 PISNR, while the efficiency reaches 100% at 30 PISNR. This is not the case for the dataset with a maximum frequency of 20 Hz, where the TAP is lower. This is expected since the sensitivity of the detectors becomes worse at lower frequencies, typically under 20 Hz, see Fig. 2. In all the figures shown in this work, the FAP is fixed at 1%.

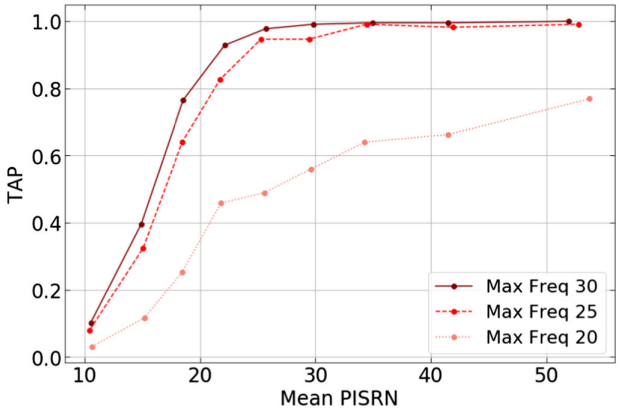

FIG. 5. The True Alarm Probability as a function of the PISRN for the O3 Gaussian noise case. Each curve represents a different test set with a different maximum frequency seen by the CNN.

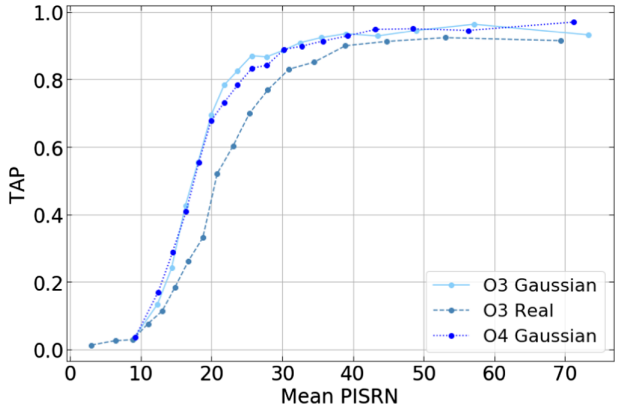

FIG. 6. The true alarm probability as a function of the PISRN for the O3 Gaussian noise, real O3 noise, and O4 Gaussian noise.

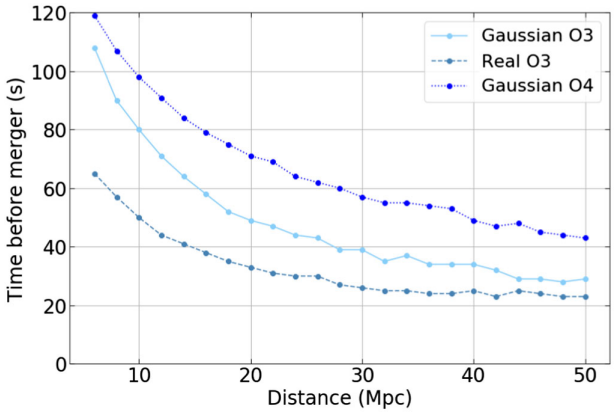

FIG. 7. The time before merger at which the event is detected as a function of the distance. These curves are made for a BNS with component masses similar to those of GW170817.

Similarly, we have done the same test for the real O3 noise and the simulated O4 Gaussian noise. The different tests are summarized in Fig. 6, where each curve represents the results for the whole test set. In terms of PISNR, the efficiencies for O3 Gaussian noise and O4 Gaussian noise are very similar. However, since the noise floor is lower in the O4 case, the network can probe higher distances in this case. The performance for real noise is a bit worse than for the two Gaussian cases. The network needs a slightly larger PISRN to achieve the same performance. For example, the network needs a PISNR of 20 to have an efficiency of 50% in the case of real O3, whereas it only needs a PISNR of 17 to reach the same sensitivity in the two other cases. Even if some glitches and non-Gaussian features are present in the data, the network is still able to reach a high performance provided that the PISNR is high enough. To be more realistic with a real time search, the noise is downloaded from strains recorded at the time of low-latency. Therefore, it has a low quality, explaining the reduced performance.

After testing the network on independent 300 s-long frames, we generate longer frames of 1000 s, and we inject a complete GW signal into them. Then, we slide a 300 s window over the frame, pass the data in the window to the CNN for each step and make a prediction. From one step to the next, the window is shifted by 5 s. This is repeated until the full 1000 s are covered. Note that the step of 5 s is arbitrary and can be reduced, since for a realistic early-alert pipeline the length of the minimum step should be equal to the time required to load 300 s of data, preprocess it, and predict it with our network. The deep-learning algorithm is fast and takes about 0.5 s on a CPU and 0.01 s on a *GeForce GTX 750 GPU*, the preprocessing is also fast: about 0.13 s to compute the PSD with PyCBC, 0.01 s to perform the whitening and 1 s to remove the peaks and do the renormalization. The limiting factor is to load 300 s of data for 3 detectors with *GWpy*,[6] which takes around 2 s on an Intel Xeon E5-2650 v4 CPU. Note that the PSD used for the whitening is computed each time we load the 300 s frame. To reduce further this time, one can compute the PSD at regular intervals and use the result for multiple steps.

Figure 7 illustrates the time left before merger when our approach is able to detect the event for the different noise types. Each point contains 1000 frames with a duration of 1000 s and each frame has a different noise realization. In each frame, we inject a BNS signal with component masses similar to those detected for GW170817 [78]. We choose fixed masses to keep the total duration of the signal fixed. The sky position of the signal is changed for each frame. We then slide a 300 s window over the 1000 s as described above. The process is then repeated for injections corresponding to a larger distance. Figure 7 shows that, for a given distance, the events are detected the earliest in O4 Gaussian noise. It is also interesting to note that the time before merger for real O3 noise and Gaussian O3 noise are similar, even if the Gaussian case is slightly better. An event like GW170817 at a distance of 40 Mpc can be detected by our method 25 s in advance in real O3 noise, 35 s in advance in Gaussian O3 noise, and 50 s in advance for Gaussian O4 noise, showing quite good trigger capabilities in future observations runs.

For an online matched filtering search, the performance is often evaluated by a false alarm rate (FAR). It represents the probability that a trigger occurs because of the noise for a given period of time [41]. The matched filtering FAR is computed for each event and represents how often the noise is expected to produce a trigger with a ranking statistic value at least as high as the one of the event. With our method, we can not compute such a FAR, but it is possible to compute, what we call the false positive over time (FP_t), which is defined as the number of false positives by a given period of time.

To compute the FP_t, we run our network over the entire O3b data using the same setup as the one described previously. We shift the observation windows by 5 s for each step, and veto the times corresponding to events

[6]Using the built-in function *gwpy.timeseries.TimeSeries.get()*.

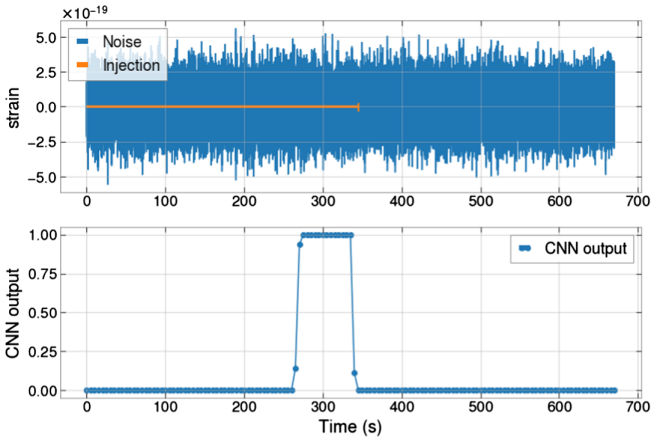

FIG. 8. Top: representation of a signal and the noise it is injected in. Bottom: representation of the output of the CNN. Each point represents the probability to have an inspiral in the 300 s of data. By convention, the time of a point represents the end of the time window. The network does not trigger on the early inspiral because the PISNR is too low. When it becomes high enough, the networks produces a trigger until the injection leaves the frame, giving multiple points with a high probability in a row.

reported in the GWTC-3 catalog [49] and assume that there are no other detectable events in the data.[7] The FP_t is then defined as the number of triggers divided by the total observation time. For O3b, we obtain a FP_t of 277.54 per day, which is too high to be used for online searches. To decrease its values, we can consider that an event is present when our network gives multiple triggers in a row, as shown in Fig. 8. If we keep detections with 5 consecutive triggers, the FP_t goes down to 12.31 per day, and it goes to 1.71 per day if we consider 10 triggers in a row. The use of multiple triggers implies larger waiting times before producing an alert, and reduces the time before merger for the detection. For example, considering 5 triggers leads to a delay of 20 s as we consider steps of 5 s when sliding the window. In the end, this shows that we would need to find a trade-off between the time before merger and the desired FP_t.

Another way to decrease the FP_t is to use coherent triggers between two or more detectors. The training strategy for the network does not make it favor coherent triggers. Indeed, since it is trained for one, two, or three detectors available, it learns to trigger even if only one detector is online. Furthermore, even if more than one detector is online, we do not use a minimum SNR in each detector for the training set. Hence, the network learns to trigger even if only one interferometer is picking up the signal. In the end, this means that as soon as the CNN sees something remotely close to a signal in one of the detectors, it triggers, leading to a relatively high FP_t.

[7]This assumption is reasonable since our network needs relatively high SNRs to detect the inspiral, and the event would therefore have been detected.

B. Number of BNS inspirals detectable in O4

To estimate the number of BNSs that our network could detect in O4, we simulate a population of BNSs. It is generated using the method described in [82] and the BNS merger rate is normalized so that the local rate is equal to the median rate given in [83]. The only difference with [82] is that we adapt the detection thresholds and the PSDs to our O4 scenario. We keep the BNS events with a network SNR higher than 13 and discard all the others. This threshold is chosen as we expect our network to find only BNSs that are clearly visible in the detector network and a global SNR of 13 corresponds approximately to an SNR of 8 in each detector.

To have more statistics, we compute the equivalent of 5 years of data, and we consider a duty cycle of 100% for all the detectors. Our simulations predict that, on average, around twenty BNSs per year will have a network SNR over 13 for O4 sensitivity. Our network can detect around three of those BNSs in advance. Figure 9 represents the time before the merger of all the BNSs detected by our network for the 5 years of generated data. Even if our network is able to detected only three events out of twenty, it is important to note that these events are seen in advance and would therefore not be seen at that stage by the unmodified matched filtering searches. Nevertheless, matched filtering pipelines adapted to the early detection of long inspirals are also being developed [18,20]. Those are also able to detect BNS mergers in advance. Even if the comparison between these works and ours is difficult (partially because of the difference in noise, but also in performance evalution), we can mention that times before merger of these algorithms are comparable to those obtained by our network, ranging from $\mathcal{O}(10)$ to $\mathcal{O}(100)$ seconds. An advantage of these early-warning matched filtering searches is that their FAR is lower than our FP_t (around one per month) but they require more computational resources during the search as the highest cost for machine learning is moved to the training step. During the search, our method can run on a single GPU or even on a

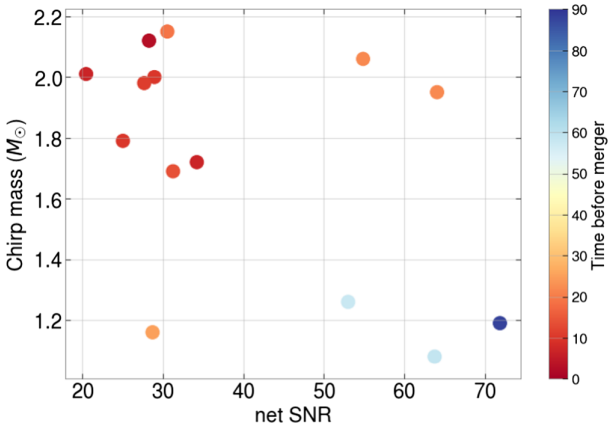

FIG. 9. The number of BNSs detected in advance by our network for a simulated population of BNSs in five years of O4 data.

TABLE II. The time before merger, the maximum frequency seen by the network at detection, the chirp mass of the event, the network SNR, and the PISNR at the moment of the detection for all the detected BNSs in five years of simulated O4 data.

Time before merger (s)	$\mathcal{M}_c$ ($M_\odot$)	Network SNR	Net PISNR at detection	Maximum frequency (Hz)
88	1.19	71.87	15.32	25.45
59	1.08	63.77	23.43	31.35
58	1.26	53.01	16.63	28.75
25	1.16	28.72	16.31	41.39
22	1.95	64.07	19.8	31.45
22	2.06	54.88	18.01	30.42
19	2.15	30.55	10.37	31.29
14	1.69	31.26	14.42	40.75
11	1.98	27.68	13.63	40.43
10	2.0	28.95	16.28	41.58
10	1.79	25.04	12.9	44.52
7	2.01	20.47	11.87	47.48
7	1.72	34.21	25.71	52.2
3	2.12	28.28	20.68	63.01

single CPU, while most of the standard matched filtering methods require parallelization on multiple CPUs.

Table II shows the different characteristics of the detected BNSs. The network can see an event when the net PISNR is between 10 and 25, which is expected according to Fig. 6. The time before the merger at which the CNN can detect a signal depends on two factors: (a) the network PISNR and (b) the length of the signal. The PISNR can be seen as a fraction of the SNR and its exact value depends on which part of the signal we are considering (hence the maximum frequency see by the CNN). Therefore, for a fixed signal duration, if the network SNR is high, the network can detect an event at a lower maximum frequency, corresponding to a longer duration before the merger. However, if we fix the SNR and the maximum frequency while increasing the duration of the signal (for example by decreasing both the chirp mass and the luminosity distance to compensate), the event will be detected with a larger duration before the merger. This behavior is well represented in Fig. 9, where events with a light chirp mass and a high SNR are detected the earliest. It also explains why some events with a lower chirp mass can be detected earlier, even if the network SNR is smaller than for other events.

IV. CONCLUSION

This work builds upon the framework developed in [14]. We implement several upgrades and modifications to the CNN-based pipeline designed to detect the early inspiral phase of BNS events. A major upgrade is the increased duration of the frames passed to the network. That allows us to search for smaller frequencies and opens the door to earlier detections. Another benefit of this increased duration is that we can use a single network to look for all type of BNSs, which was not the case in our previous work. The detection of events for a smaller maximum frequency is not easy and required an adapted training methodology: curriculum learning. We consider realistic observation scenarios, including all the detectors of the LIGO-Virgo network and use realistic noise realizations: O3 and O4 Gaussian noises, as well as real O3 noise. We have also demonstrated that even in the real O3 noise our network is able to detect GW signals in advance. We expect our network to detect some BNSs in O4, up to minutes in advance if the SNR of the event is high enough. In future works, we will upgrade our method to search for neutron-star-black-hole mergers as well. As discussed in Sec. III, we will also develop methods to decrease the FP_t. Finally, we will investigate a way to infer the sky position with only the early inspiral part.

ACKNOWLEDGMENTS

The authors thank Thomas Dent and Srashti Goyal for their useful comments, as well as Maxime Fays, Vincent Boudart, Sarah Caudill and Chris Van Den Broeck for useful discussions. G. B. is supported by a FRIA grant from the Fonds de la Recherche Scientifique-FNRS, Belgium. J. R. C. acknowledges the support of the Fonds de la Recherche Scientifique-FNRS, Belgium, under Grant No. 4.4501.19. M. L. H. N., and J. J. are supported by the research program of the Netherlands Organisation for Scientific Research (NWO). The authors are grateful for computational resources provided by the LIGO Laboratory and supported by the National Science Foundation Grants No. PHY-0757058 and No. PHY-0823459. 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: DETAILS ON THE VETO OF PEAKS FOR REAL NOISE

After the noise has been downloaded and whitened, large peaks can still be present in the data (see Fig. 10).

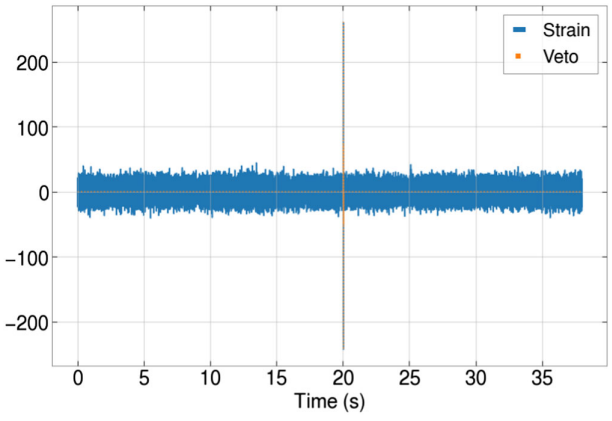

FIG. 10. The blue curve represents O3a noise after application of the whitening, the low-pass filter, and the high-pass filter. The orange curve shows the part which will be vetoed.

This behavior only appears for real O3 noise, and leads to a problem for the normalization. Indeed, before passing the data to the network, we normalize them to be between −1 and 1. To do so, we find the maximum absolute value of the strain and divided each point by that value. When a large peak is present, the maximum absolute value is the value of the peak and it makes the rest of the time series too small. That confuses the neural network, and we decided to veto these peaks. The vetoing is done according to the z-score, which is defined as:

$$Z_i = \frac{x_i - \mu}{\sigma} \tag{A1}$$

where x_i is the value of a point i in the time series, μ and σ are respectively the mean and the variance of the time series. We then compute the standard deviation of the z-score and put all the points with a z-score larger than 5 times the standard deviation to zero, allowing to remove large peaks such as those seen in Fig. 10. The normalization is then done on the vetoed frame.

[1] P. Mészáros, D. B. Fox, C. Hanna, and K. Murase, Nat. Rev. Phys. **1**, 585 (2019).
[2] W. Arnett, J. N. Bahcall, R. P. Kirshner, and S. E. Woosley, Annu. Rev. Astron. Astrophys. **27**, 629 (1989).
[3] S. Ansoldi *et al.* (MAGIC Collaboration), Astrophys. J. Lett. **863**, L10 (2018).
[4] J. Goodman, Astrophys. J. **308**, L47 (1986).
[5] J. Aasi *et al.* (LIGO Scientific Collaboration), Classical Quantum Gravity **32**, 074001 (2015).
[6] F. Acernese *et al.* (VIRGO Collaboration), Classical Quantum Gravity **32**, 024001 (2015).
[7] T. Akutsu *et al.* (KAGRA Collaboration), Nat. Astron. **3**, 35 (2019).
[8] K. Cannon *et al.*, Astrophys. J. **748**, 136 (2012).
[9] L. P. Singer *et al.*, Astrophys. J. **795**, 105 (2014).
[10] C. Meegan *et al.*, Astrophys. J. **702**, 791 (2009).
[11] A. Goldstein *et al.*, Astrophys. J. **848**, L14 (2017).
[12] B. P. Abbott *et al.*, Astrophys. J. Lett. **848**, L12 (2017).
[13] A. Perego, D. Radice, and S. Bernuzzi, Astrophys. J. Lett. **850**, L37 (2017).
[14] G. Baltus, J. Janquart, M. Lopez, A. Reza, S. Caudill, and J.-R. Cudell, Phys. Rev. D **103**, 102003 (2021).
[15] S. Sachdev *et al.*, Astrophys. J. Lett. **905**, L25 (2020).
[16] H. Yu, R. X. Adhikari, R. Magee, S. Sachdev, and Y. Chen, Phys. Rev. D **104**, 062004 (2021).
[17] T. Tsutsui, A. Nishizawa, and S. Morisaki, Mon. Not. R. Astron. Soc. **512**, 3878 (2022).
[18] A. H. Nitz, M. Schäfer, and T. D. Canton, Astrophys. J. Lett. **902**, L29 (2020).
[19] A. H. Nitz, T. D. Canton, D. Davis, and S. Reyes, Phys. Rev. D **98**, 024050 (2018).
[20] S. Sachdev *et al.*, Astrophys. J. Lett. **905**, L25 (2020).
[21] M. L. Chan, C. Messenger, I. S. Heng, and M. Hendry, Phys. Rev. D **97**, 123014 (2018).
[22] Y. Li, I. S. Heng, M. L. Chan, C. Messenger, and X. Fan, Phys. Rev. D **105**, 043010 (2022).
[23] M. Kovalam, M. A. Kaium Patwary, A. K. Sreekumar, L. Wen, F. H. Panther, and Q. Chu, Astrophys. J. Lett. **927**, L9 (2022).
[24] R. Magee and S. Borhanian, arXiv:2201.11841.
[25] L. Nuttall, and C. P. L. Berry, Astron. Geophys. **62**, 4.15 (2021).
[26] R. Magee *et al.*, Astrophys. J. Lett. **910**, L21 (2021).
[27] M. Saleem, A. Pai, K. Misra, L. Resmi, and K. G. Arun, Mon. Not. R. Astron. Soc. **475**, 699 (2018).
[28] D. Reitze *et al.*, Bull. Am. Astron. Soc. **51**, 035 (2019), https://baas.aas.org/pub/2020n7i035/release/1.
[29] M. Punturo *et al.*, Classical Quantum Gravity **27**, 194002 (2010).
[30] A. H. Nitz and T. Dal Canton, Astrophys. J. Lett. **917**, L27 (2021).
[31] C. Messick *et al.*, Phys. Rev. D **95**, 042001 (2017).
[32] S. Sachdev *et al.*, arXiv:1901.08580.
[33] C. Hanna *et al.*, Phys. Rev. D **101**, 022003 (2020).
[34] K. Cannon, S. Caudill, C. Chan, B. Cousins, J. D. Creighton, B. Ewing, H. Fong, P. Godwin, C. Hanna, S. Hooper *et al.*, SoftwareX **14**, 100680 (2021).
[35] B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown, and J. D. E. Creighton, Phys. Rev. D **85**, 122006 (2012).
[36] B. Allen, Phys. Rev. D **71**, 062001 (2005).
[37] T. Dal Canton *et al.*, Phys. Rev. D **90**, 082004 (2014).
[38] S. A. Usman *et al.*, Classical Quantum Gravity **33**, 215004 (2016).
[39] A. H. Nitz, T. Dent, T. D. Canton, S. Fairhurst, and D. A. Brown, Astrophys. J. **849**, 118 (2017).
[40] G. S. Davies, T. Dent, M. Tápai, I. Harry, C. McIsaac, and A. H. Nitz, Phys. Rev. D **102**, 022004 (2020).
[41] T. Adams, D. Buskulic, V. Germain, G. M. Guidi, F. Marion, M. Montani, B. Mours, F. Piergiovanni, and G. Wang, Classical Quantum Gravity **33**, 175012 (2016).
[42] F. Aubin *et al.*, Classical Quantum Gravity **38**, 095004 (2021).
[43] S. Klimenko, G. Vedovato, M. Drago, F. Salemi, V. Tiwari, G. A. Prodi, C. Lazzaro, K. Ackley, S. Tiwari, C. F. Da Silva, and G. Mitselmakher, Phys. Rev. D **93**, 042004 (2016).
[44] S. Klimenko and G. Mitselmakher, Classical Quantum Gravity **21**, S1819 (2004).
[45] S. Klimenko, G. Vedovato, M. Drago, G. Mazzolo, G. Mitselmakher, C. Pankow, G. Prodi, V. Re, F. Salemi, and I. Yakushin, Phys. Rev. D **83**, 102001 (2011).
[46] Q. Chu *et al.*, Phys. Rev. D **105**, 024023 (2022).
[47] B. P. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Astrophys. J. **875**, 161 (2019).
[48] R. Abbott *et al.*, arXiv:2108.01045.
[49] R. Abbott *et al.* (LIGO Scientific, VIRGO, and KAGRA Collaborations), arXiv:2111.03606.

[50] D. Davis *et al.*, Classical Quantum Gravity **38**, 135014 (2021).
[51] I. Goodfellow *et al.*, *Deep Learning* (MIT Press, Cambridge, MA, 2016), http://www.deeplearningbook.org.
[52] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).
[53] T. Gebhard *et al.*, in *Workshop on Deep Learning for Physical Sciences (DLPS) at the 31st Conference on Neural Information Processing Systems (NIPS)* (Curran Associates Inc., Red Hook, 2017), pp. 1–6.
[54] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Phys. Rev. D **100**, 063015 (2019).
[55] P. G. Krastev, Phys. Lett. B **803**, 135330 (2020).
[56] M. B. Schäfer, F. Ohme, and A. H. Nitz, Phys. Rev. D **102**, 063015 (2020).
[57] P. Astone, P. Cerdá-Durán, I. Di Palma, M. Drago, F. Muciaccia, C. Palomba, and F. Ricci, Phys. Rev. D **98**, 122002 (2018).
[58] A. Iess, E. Cuoco, F. Morawski, and J. Powell, Mach. Learn. Sci. Tech. **1**, 025014 (2020).
[59] M. López, I. Di Palma, M. Drago, P. Cerdá-Durán, and F. Ricci, Phys. Rev. D **103**, 063011 (2021).
[60] V. Boudart, and M. Fays, Phys. Rev. D **105**, 083007 (2022).
[61] S. R. Green, C. Simpson, and J. Gair, Phys. Rev. D **102**, 104057 (2020).
[62] M. J. Williams, J. Veitch, and C. Messenger, Phys. Rev. D **103**, 103006 (2021).
[63] A. Kolmus *et al.*, Phys. Rev. D **106**, 023032 (2022).
[64] M. Zevin *et al.*, Classical Quantum Gravity **34**, 064003 (2017).
[65] S. Soni *et al.*, Classical Quantum Gravity **38**, 195016 (2021).
[66] J. McGinn, C. Messenger, M. J. Williams, and I. S. Heng, Classical Quantum Gravity **38**, 155005 (2021).
[67] M. Lopez *et al.*, Phys. Rev. D **106**, 023027 (2022).
[68] E. Cuoco *et al.*, Mach. Learn. Sci. Tech. **2**, 011002 (2021).
[69] Y. Bengio *et al.*, in *Proceedings of the 26th Annual International Conference on Machine Learning* (Association for Computing Machinery, New York, 2009), pp. 41–48.
[70] C. Cutler and E. E. Flanagan, Phys. Rev. D **49**, 2658 (1994).
[71] C. M. Biwer, C. D. Capano, S. De, M. Cabero, D. A. Brown, A. H. Nitz, and V. Raymond, Publ. Astron. Soc. Pac. **131**, 024503 (2019).
[72] W. G. Anderson and J. D. E. Creighton, in *Short-Period Binary Stars: Observations, Analyses, and Results*, Astrophysics and Space Science Library Vol. 352 (Springer, Dordrecht, 2008), 10.1007/978-1-4020-6544-6_2.
[73] B. P. Abbott *et al.*, Living Rev. Relativity **23**, 3 (2020).
[74] D. M. Macleod, J. S. Areeda, S. B. Coughlin, T. J. Massinger, and A. L. Urban, SoftwareX **13**, 100657 (2021).
[75] P. Petrov, L. P. Singer, M. W. Coughlin, V. Kumar, M. Almualla, S. Anand, M. Bulla, T. Dietrich, F. Foucart, and N. Guessoum, Astrophys. J. **924**, 54 (2022).
[76] A. Buonanno, Y. Chen, and M. Vallisneri, Phys. Rev. D **67**, 104025 (2003); **74**, 029904(E) (2006).
[77] B. Kiziltan, A. Kottas, M. De Yoreo, and S. E. Thorsett, Astrophys. J. **778**, 66 (2013).
[78] B. P. Abbott *et al.*, Phys. Rev. Lett. **119**, 161101 (2017).
[79] D. Kingma and J. A. Ba, arXiv:1412.6980.
[80] P. T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, Ann. Oper. Res. **134**, 19 (2005).
[81] A. Paszke *et al.*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library* (Curran Associates, Inc., Red Hook, 2019), pp. 8024–8035, http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
[82] A. Samajdar, J. Janquart, C. V. Den Broeck, and T. Dietrich, Phys. Rev. D **104**, 044003 (2021).
[83] R. Abbott *et al.* (LIGO Scientific and Virgo Collaborations), Astrophys. J. Lett. **913**, L7 (2021).