# Deep Neural Networks to Enable Real-time Multimessenger Astrophysics

Daniel George[1,2] and E. A. Huerta[2]

[1]*Department of Astronomy, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801, USA*
[2]*NCSA, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801, USA*

Gravitational wave astronomy has set in motion a scientific revolution. To further enhance the science reach of this emergent field, there is a pressing need to increase the depth and speed of the gravitational wave algorithms that have enabled these groundbreaking discoveries. To contribute to this effort, we introduce `Deep Filtering`, a new highly scalable method for end-to-end time-series signal processing, based on a system of two deep convolutional neural networks, which we designed for *classification* and *regression* to rapidly detect and estimate parameters of signals in highly noisy time-series data streams. We demonstrate a novel training scheme with gradually increasing noise levels, and a transfer learning procedure between the two networks. We showcase the application of this method for the *detection* and *parameter estimation* of gravitational waves from binary black hole mergers. Our results indicate that `Deep Filtering` significantly outperforms conventional machine learning techniques, achieves similar performance compared to matched-filtering while being several orders of magnitude faster thus allowing real-time processing of raw big data with minimal resources. More importantly, `Deep Filtering` extends the range of gravitational wave signals that can be detected with ground-based gravitational wave detectors. This framework leverages recent advances in artificial intelligence algorithms and emerging hardware architectures, such as deep-learning-optimized GPUs, to facilitate real-time searches of gravitational wave sources and their electromagnetic and astro-particle counterparts.



## I. INTRODUCTION

GW概述之近况

Gravitational wave (GW) astrophysics is by now a well established field of research. The advanced Laser Interferometer Gravitational wave Observatory (aLIGO) detectors have detected four significant GW events, consistent with Einstein's general relativity predictions of binary black hole (BBH) mergers [1–4]. These major scientific breakthroughs, worthy of the 2017 Nobel Prize in Physics, have initiated a new era in astronomy and astrophysics. By the end of aLIGO's second discovery campaign, referred to as O2, the European advanced Virgo detector [5] joined aLIGO, establishing the first, three-detector search for GW sources in the advanced detector era. We expect that ongoing improvements in the sensitivity of aLIGO and Virgo within the next few months will continue to increase the number and types of GW sources.

GW astrophysics is a multidisciplinary enterprise. Experimental and theoretical physics, cosmology, fundamental physics, high performance computing (HPC) and high throughout computing have been combined into a coherent program to revolutionize our understanding of the Universe. For instance, at the interface of HPC and theoretical physics, numerical relativity (NR) simulations of Einstein's field equations are extensively used to validate the astrophysical nature of GW transients [6]. Furthermore, NR simulations of binary neutron star (BNS) mergers, neutron star-black hole (NSBH) mergers, core collapse supernovae and other massive, relativistic systems provide key physical insights into the physics of systems that are expected to generate electromagnetic (EM) and astro-particle counterparts [7–12].

GW是多学科的

Ongoing discovery campaigns with GW detectors and astronomical facilities [13–18] have already led to *multimessenger* observations of GW events and their EM counterparts [19–21]. These complementary observations have provided new and detailed information about the astrophysical origin, and cosmic evolution of ultra compact objects [7, 22–27]. The time sensitive nature of these analyses requires algorithms that can detect and characterize GW events in real-time [28].

GW的multimessenger发展

介绍MF及其参数多维度估测的目标

aLIGO's flagship matched-filtering searches have been very successful at identifying and characterizing GW transients [29–32]. Looking ahead in the near future, GW discovery campaigns will be longer, and data will be gathered by a network of interferometers in several continents. In anticipation for this scenario, LIGO scientists now exploit state-of-the-art HPC facilities to increase the pool of computational resources to carry out for large scale GW data analysis. To maximize the science we can extract from GW observations, it is essential to cover a deeper parameter space of astrophysically motivated sources, i.e., we need to increase the dimensionality of existing GW searches from 3-dimensions (3D) to 9D[1]. Furthermore, accelerating parameter estimation algorithms, which typically last from several hours to a few days, is no trivial task since they have to sample a 15D parameter space [33]. This is a grand computational challenge given the compute intensive nature of large scale GW searches [34].

To start addressing these pressing issues, we introduce `Deep Filtering`, a new machine (deep) learning algorithm, based on deep neural networks (DNNs) [35] to directly process highly noisy time-series data for both classification and regression. `Deep Filtering` consists of two deep convolutional neural networks [36] that directly take time-series inputs and are capable of detecting and characterizing signals whose peak power is significantly weaker than that of the background noise. In this foundational article, we carry out a systematic assessment of DNNs trained to cover the stellar-mass, BBH parameter-space, where ground-based GW detectors are expected to have the highest detection rate [37]. As a first step, to construct and validate `Deep Filtering`,

介绍Deep Filtering
+说明BBH波形的来源

[1] 9D: component masses, eccentricity, and two (3D) vectors describing the spin of each binary component.

we have used a dataset of inspiral-merger-ringdown (IMR) BBH waveforms for training [38].

As discussed in [34], the computational cost of matched-filtering searches increases significantly when targeting GW sources that span a higher dimensional parameter space. In contrast, when using deep learning, all the intensive computation is diverted to the one-time training stage, after which the datasets can be discarded, i.e., the size of template banks that describe the GW signals we search for present no limitation when using deep learning. Indeed, it is preferable to use large datasets of GW signals for the one-time training stage to cover as deep a parameter space as possible. With existing computational resources on supercomputers such as Blue Waters, we estimate that it would possible to finish training the DNNs on templates across 10 or more dimensions of parameters within a few weeks.

The main objective in developing `Deep Filtering` is to enhance existing, low latency GW detection algorithms to enable deeper and faster GW searches. We envision using `Deep Filtering` to identify and rapidly constrain the astrophysical parameters of GW transients. This real-time analysis would then be followed up by existing LIGO pipelines focusing on a narrow region of GWs' higher dimensional parameter space. A targeted search of this nature will significantly reduce the size of multi-dimensional template banks, enabling the use of established matched-filtering searches at a fraction of their computational cost to quantify the significance of new GW detections. This approach would combine the best of two approaches: the scalable, multidimensional nature of neural networks with the sophistication of LIGO detection pipelines. To accomplish this, we are working with the developers of `PyCBC` [29] to implement `Deep Filtering` as a module to increase the depth and speed of this pipeline.

The results we present in this article confirm that DNNs are ideal tools for future GW analysis. We have found that DNN are able to *interpolate* between waveform templates, in a similar manner to Gaussian Process Regression (GPR) [2], and to *generalize* to new classes of signals beyond the templates used for training. Furthermore, our DNNs can be evaluated faster than real-time with a single CPU, and very intensive searches over a broader range of signals can be easily carried out with one dedicated GPU. The intelligent nature of deep learning would allow automated learning of persistent and transient characteristics of noises inherent to the detectors, while incorporating real-time data quality information. This analysis, combined with recent work to understand and characterize aLIGO non-Gaussian noise transients [42, 43], strongly suggests that it is feasible to create a single efficient pipeline to perform all tasks—identifying the presence or absence of GW signals, classifying noise transients, and reconstructing the astrophysical properties of detected GW sources. Furthermore, since this technique can be applied to other types of raw time-series data, similar DNNs can be used to process telescope data, thus paving a natural path to realizing real-time multi-messenger astrophysics with a unified framework.

As NR continues to shed light into the physics of GW sources[6], we will rely on an extensive exploitation of HPC resources to obtain NR waveforms to train our DNN algorithm. At the same time, we are using HPC facilities to carry out large scale parameter sweeps to find optimal DNNs for GW detection and parameter estimation. The approach we discuss here employs recent advances in artificial intelligence algorithms, by computer scientists and industries, for accelerating scientific discovery by enhancing the use of traditional HPC resources, while allowing us to exploit emerging hardware architectures such as deep-learning-optimized Graphics Processing Units (GPUs) [44], Application-Specific Integrated Circuits (ASICs) [45], Field-Programmable Gate Arrays (FPGAs) [46], quantum computers [47] and brain-like neuromorphic chips [48]. This approach may provide the needed platform to address common challenges on large scale data analytics on disparate fields of research to effectively consolidate different windows of observation into the Universe.

This article is organized as follows: Section II provides a comprehensive overview of artificial neural networks and deep learning, particularly focusing on convolutional neural networks in the context of time-series signal processing. In Section III, we describe our assumptions, datasets, and procedure to construct the DNN-based GW analysis pipeline. We report the results of our analysis in Section IV. In Section V, we discuss its immediate applications, and their implications for GW astrophysics missions, along with scope for improvements. We summarize our findings and outline its broader impact in Section VI.

## II. DEEP NEURAL NETWORKS

In this section we provide a brief overview of the main concepts of deep learning, including machine learning, artificial neural networks, and convolutional neural networks in the context of time-series signal processing.

The vast majority of algorithms are designed with a specific task in mind. They require extensive modifications before they can be re-used for any other task. The term machine learning refers to a special class of algorithms that can *learn* from examples to solve new problems without being explicitly re-programmed. This enables cross-domain applications of the same algorithm by training it with different data [49]. More importantly, some of these algorithms are able to tackle problems which humans can solve intuitively but find difficult to explain using well-defined rules, hence they are often called "artificial intelligence" [49].

The two main categories of machine learning are supervised and unsupervised learning. In supervised learning, the algorithm learns from some data that is correctly labeled, while unsupervised learning algorithms have to make sense of unstructured and unlabeled data [50]. We will be focusing on an

[2] GPR [39–41] is a statistical tool that can serve as a probabilistic interpolation algorithm providing information about the training set of NR simulations needed to accurately describe a given parameter-space and generates interpolated waveforms that match NR counterparts above any given accuracy.

巨大计算开销, DL可以避免

目标: 更深, 更快的GW search (联合MF)并入到pipeline.

结果启示: GW模板可推广. (对non-Gaussion noise研究的启示)

DNN还能做估测~

不同的硬件框架

ML AI

ML=supervised + unsupervised

application of supervised learning in this work, where we use labeled data obtained from physics simulations to train an algorithm to detect signals embedded in noise and also estimate multiple parameters of the source.

limit of ML

Although traditional machine learning algorithms have been successful in several applications, they are limited in their ability to deal directly with raw data. Often the data has to be simplified manually into a representation suitable for each problem. Determining the right representation is extremely difficult and time-consuming, often requiring decades of effort even for domain experts, which severely limits the applicability of these algorithms [49].

representation learning => DL

Representation learning is a subset of machine learning which aims to resolve this issue by creating algorithms that can learn by themselves to find useful representations of the raw data and extract relevant features from it automatically for each problem [51]. Here, we are focusing on a special type of representation learning called deep learning.

### Deep Learning

What DL?

Deep learning is a new subfield of machine learning, which resolves this difficulty of feature engineering with algorithms that learn by themselves to find useful representations of the raw data, and extract multiple levels of relevant features from it automatically for each problem. This is achieved by combining a computational architecture containing long interconnected layers of "artificial neurons" with powerful learning (optimization) algorithms [35, 49]. These deep artificial neural networks (DNNs) are able to capture complex non-linear relationships in the data by composing hierarchical internal representations, all of which are learned automatically during the training stage. The deepest layers are able to learn highly abstract concepts, based on the simpler outputs of the previous layers, to solve problems that previously required human-level intelligence [50].

How DL now?

Various factors including the exponential growth of computational resources (especially GPUs), availability of massive amounts of data, and the development of new algorithmic techniques and software have recently contributed to make deep learning very successful in commercial applications, thus revolutionizing multiple industries today. The state-of-the-art algorithms for image processing, speech recognition, natural language understanding are all based on deep learning. DNNs power many of the technologies routinely used by us including search engines (Google, Bing), voice recognition, personal assistants (Siri, Cortana, Google assistant), text prediction on mobile keyboards, real-time face detection on cameras, face recognition (e.g. face-tagging in Facebook), language translation (Google Translate), text-to-speech synthesis [52], recommendations on Amazon, and automatic captioning on YouTube, to name a few [53].

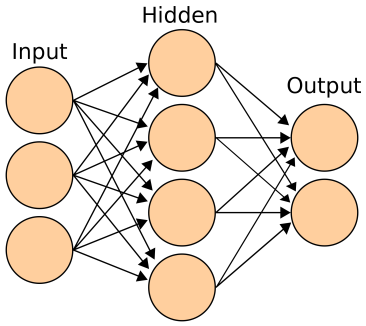


FIG. 1. An Artificial Neural Network (ANN) or multilayer perceptron with one hidden layer is depicted [60]. The circles represent neurons and arrows represent connections (weights) between neurons. Note that each neuron has only a *single* output, which branches out to connect with neurons in the next layer.

### Artificial Neural Networks

What ANN & perce-ptrons?

Artificial neural networks (ANN), the building blocks of DNNs, are biologically-inspired computational models that have the capability to learn from observational data [54]. The fundamental units of neural networks are artificial neurons (loosely modeled after real neurons [55]), which are based on perceptrons introduced by Rosenblatt in 1957 [56]. A perceptron takes a vector of inputs ($\vec{x}$) and computes a weighted output with an offset known as bias. This can be modeled by the equation $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$, where the weights ($\vec{w}$) and bias ($b$) are learned through training.

multi. perce-ptrons

Minsky and Papert showed in their 1969 book Perceptrons [57] that a single perceptron has many limitations. Unfortunately, this led to a decline in the popularity of all neural networks in the following decades [50]. However, it was later found that these limitations can be overcome by using multiple layers of inter-connected perceptrons to create ANNs. The universality theorem [58] proves that ANNs with just three layers (one hidden layer) can model any function up to any desired level of accuracy.

not RNN

Multilayer perceptrons are also known as feed-forward neural networks because information is propagated forward from the input layer to the output layer without internal cycles (i.e no feedback loops) [49]. While potentially more powerful cyclic architectures can be constructed, such as Recurrent Neural Networks (RNNs), we will be focusing mainly on simple feed-forward neural networks in this article.

non-linearity (ramp)

An ANN usually has an input layer, one or more hidden layers, and an output layer (shown in Figure 1). A non-linear "activation" function is applied to the output of each of the hidden layers. Without this non-linearity, using multiple layers would become redundant, as the network will only be able to express linear combinations of the input. The most commonly used non-linear activation functions are the logistic sigmoid, hyperbolic tan, and the rectified linear unit (also called ReLU or ramp). It has been empirically observed that the ramp produces the best results for most applications [59] . This function is mathematically expressed as $max(0, x)$.

Learnning: BP+ error+ SGD+ ADAM

The key ingredient that makes ANNs useful is the learning algorithm. Almost all neural networks used today are trained with variants of the back-propagation algorithm based on the steepest descent method [50]. The idea is to propagate errors backward from the output layer to the input layer after each evaluation of a neural network, in order to adjust the weights of each neuron so that the overall error is reduced in a supervised learning problem [61]. The weights of an ANN are usually initialized randomly to small values and then back-propagation is performed over multiple rounds, known as epochs, until the errors are minimized. Stochastic gradient descent with mini-batches [62] has been the traditional method used for back-propagation. This technique uses an estimate of the gradient of the error over subsets of the training data in each iteration to change the weights of the ANN. The magnitude of these changes is determined by the "learning rate". New methods with variable learning rates such as ADAM (Adaptive Momentum Estimation) are becoming more popular and have been shown empirically to achieve better results more quickly [63].

### Convolutional Neural Networks

History of CNN

A convolutional neural network (CNN), whose structure is inspired by studies of the visual cortex in mammals [49], is a type of feed-forward neural network. First developed by Fukushima for his Neocognitron [64], they were successfully combined with back-propagation by LeCun [36] in the 1980s, for developing a highly accurate algorithm for recognizing handwritten digits. The exceptional performance of Alex Krizhevsky's entry based on CNNs, which won the ImageNet competition by a huge margin in 2012 [65], has sparked the current interest in these networks especially in the field of computer vision. CNNs have been most effective for image and video processing. They have been shown to approach or even surpass human-level accuracy at a variety of constrained tasks such as hand-writing recognition, identifying objects in photos, tracking movements in videos etc. [35].

Intro. Conv. layer

The introduction of a "convolution layer", containing a set of neurons that share their weights, is the critical component of these networks. Multiple convolution layers are commonly found in DNNs, with each having a separate set of shared weights that are learned during training. The name comes from the fact that an output equivalent to a convolution, or sometimes cross-correlation [49], operation is computed with a kernel of fixed size. A convolutional layer can also be viewed as a layer of identical neurons that each "look" at small overlapping sections of the input, defined as the receptive field.

Adv. of CNNs

The main advantage of using these layers is the ability to reduce computational costs by having shared weights and small kernels, thus allowing deeper networks and faster training and evaluation speeds. Because of the replicated structure, CNNs are also able to automatically deal with spatially translated as well as (with a few modifications [35]) rotated and scaled signals. In practice, multiple modules each consisting of a sequence of convolution and pooling (sub-sampling) layers, followed by a non-linearity, are used. The pooling layers further reduces computational costs by constraining the size of the DNN, while also making the networks more resilient to noise and translations, thus enhancing their ability to handle new inputs [35]. Dilated convolutions [66] is a recent development which enables rapid aggregation of information over larger regions by having gaps within each of the receptive fields. In this study, we focus on CNNs as they are the most efficient DNNs on modern hardware, allowing fast training and evaluation (inference).

### Time-series Analysis with Convolutional Neural Networks

M.F limited

Conventional methods of digital signal processing such as matched-filtering (cross-correlation or convolution against a set of templates) [67] in time-domain or frequency-space are limited in their ability to scale to a large parameter-space of signal templates, while being too computationally intensive for real-time parameter estimation analysis [33]. Signal processing using machine learning in the context of GW astrophysics is an emerging field of research [42, 68–73]. These traditional machine learning techniques, including shallow ANNs, require "handcrafted" features extracted from the data as inputs rather than the raw noisy data itself. DNNs, on the other hand, are capable of extracting these features automatically.

2D CNNs not optimal

Deep learning has been previously applied only for the classification of glitches with spectrogram images as inputs to CNNs [43, 74, 75] and unsupervised clustering of transients [43], in the context of aLIGO. Using images as inputs is advantageous for two reasons: (i) there are well established architectures of 2D CNNs which have been shown to work (GoogLeNet [76], VGG [77], ResNet [78]) and (ii) pre-trained weights are available for them, which can significantly speed up the training process via transfer learning while also providing higher accuracy even for small datasets [43]. However, our experiments showed that this approach would not be optimal for detection or parameter estimation since many signals having low signal-to-noise ratio (SNR [3]) are not visible in spectrograms, as shown in Fig. 2.

Still time-series

Theoretically, all the information about the signal is encoded within the time-series, whereas spectrograms are lossy non-invertible representations of the original data. Although 2D CNNs are commonly used, especially for image-related tasks, we found that by directly feeding raw time-series data as inputs to certain types of CNNs, one can obtain much higher sensitivities at low SNR, significantly lower error rates in parameter estimation, and faster analysis speeds. This automated feature learning allows the algorithm to develop more optimal strategies of signal processing than when given hand-extracted information such as spectrograms. There has only

[3] Note that we are using the standard definition of optimal matched-filtering SNR, as described in [79]. This SNR is on average proportional to $12.9 \pm 1.4$ times the ratio of the amplitude of the signal to the standard deviation of the noise for our test set.

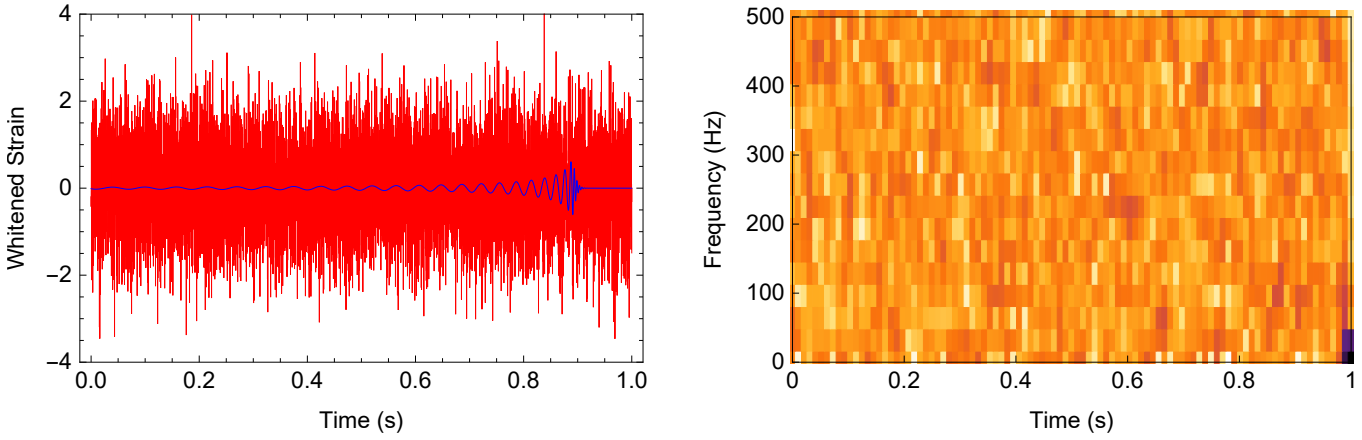


FIG. 2. **Sample of input data**. The red time-series is an example of the input to our DNN algorithm. It contains a BBH GW signal (blue) which was whitened with aLIGO's design sensitivity and superimposed in noisy data with SNR = 7.5 (peak power of this signal is 0.36 times the power of background noise). The component masses of the merging BHs are $57M_\odot$ and $33M_\odot$. The corresponding spectrogram on the right shows that the GW signal on the left is not visible, and thus cannot be detected by an algorithm trained for image recognition. Nevertheless, our DNN detects the presence of this signal directly from the (red) time-series input with over 99% sensitivity and reconstructs the source's parameters with a mean relative error of about 10%.

been a few attempts at signal processing using CNNs with raw time-series data in general and only for single parameter estimation [80, 81].

In this work, we demonstrate, for the first time, that DNNs can be used for both signal detection and multiple-parameter estimation directly from highly noisy time-series data, once trained with templates of the expected signals, and that dilated CNNs outperform traditional machine learning algorithms, and reach accuracies comparable to matched-filtering methods. We also show that our algorithm is far more computationally efficient than matched-filtering. Instead of repeatedly performing overlap computations against all templates of known signals, our CNN builds a deep *non-linear* hierarchical structure of nested convolutions, with small kernels, that determines the parameters in a single evaluation. Moreover, the DNNs act as an efficient compression mechanism by learning patterns and encoding all the relevant information in their weights, analogous to a reduced-order model [82], which is significantly smaller than the size of the training templates. Therefore, the DNNs automatically perform an internal optimization of the search algorithm and can also interpolate, or even extrapolate, to new signals not included in the template bank (unlike matched-filtering).

Note that matched-filtering is equivalent to a single convolution layer of a neural network, with very long kernels corresponding to all the signals in a template bank. Therefore, our algorithm can be viewed as an extension of matched-filtering, which performs template matching against a small set of short duration templates, and aggregates this information in the deeper layers to effectively model the full range of long-duration signals.

## III. METHOD

Our goal is to show that `Deep Filtering` is a powerful tool for GW data analysis. We do this by demonstrating that a system of two DNNs can *detect* and *characterize* GW signals embedded in highly noisy time-series data.

As a proof of concept, we focus on GWs from BBH mergers, which are expected to dominate the number of GW detections with ground-based GW detectors [3, 37, 83]. In future work, we will extend this method to signals produced by other events by adding more neurons in the final layer and training with larger datasets.

For BBH mergers

We chose to divide the problem into two separate parts, each assigned to a different DNN. The first network, henceforth known as the "classifier", will detect the presence of a signal in the input, and will provide a confidence level for the detection. The classes chosen for now are "True" or "False" depending on whether or not a signal from a BBH merger is present in the input. The second network, which we call the "predictor", will estimate the parameters of the source of the signal (in this case, the component masses of the BBH). The predictor is triggered when the classifier identifies a signal with a high probability.

We partitioned the system in this manner so that, in the future, more classes of GW transients [8, 9, 84], may be added to the classifier, and separate predictors can be made for each type of signal. Moreover, categories for various types of anomalous sources of noise, like glitches and blips [32, 74], can also be added to the classifier [43].

Future: other noises~

### Assumptions

For this initial study, we have assumed the signals are optimally oriented with respect to the detectors, and that the in-

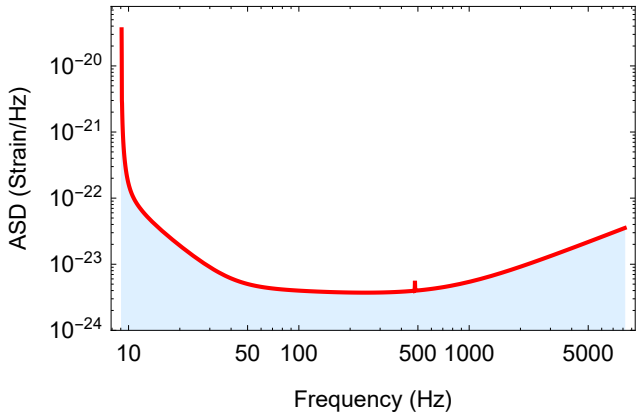


FIG. 3. Throughout this analysis, we have used the Zero Detuned High Power sensitivity configuration for aLIGO [85] to simulate the colored noise in the detectors.

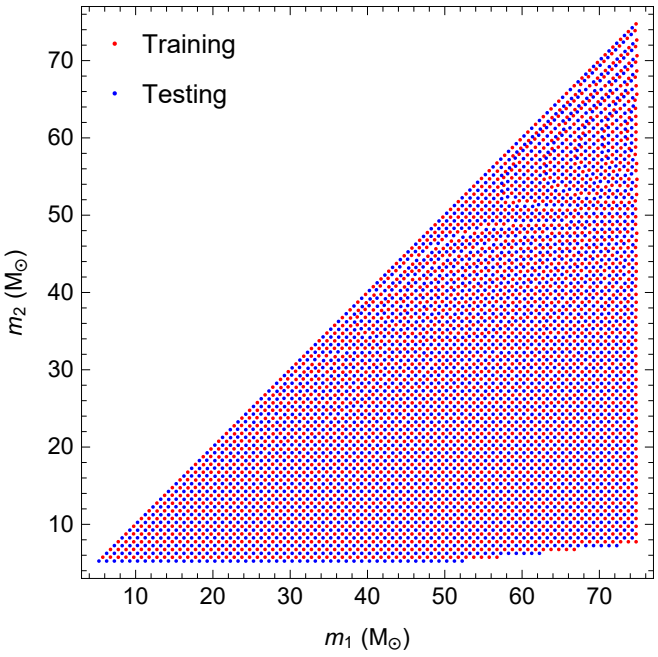


FIG. 4. **Distribution of data.** The figure shows the distribution of component masses of BBHs for the training and testing datasets. The mass-ratios were confined between 1 and 10, which accounts for the missing points in the lower right corner. We choose this mass-ratio range because the state-of-the-art EOB model we have used to create the datasets has only been validated for these mass-ratio values. Each point represents a quasi-circular, non-spinning GW signal of 1 second duration, sampled at 8192 Hz, which is whitened with aLIGO's expected noise spectrum at design sensitivity. These waveforms were normalized and translated randomly in time. Thereafter, multiple batches of noise at each SNR were added to produce training and testing datasets.

Param. of GW waveform

dividual spins and orbital eccentricities are zero. This reduces our parameter space to two dimensions, namely, the individual masses of the BBH systems, which we have restricted to lie between $5M_\odot$ and $75M_\odot$. Furthermore, we have constrained the inputs to have a duration of 1 second, and a sampling rate of 8192Hz throughout this analysis, which is more than sufficient for the events we are considering. Note that the classifier will be applied to the continuous data stream by using a sliding window of width 1 second.

Noise we used

Ideally, the inputs to our DNNs will be the unprocessed time-series of strains measured by the GW detectors. Throughout this analysis, however, we have whitened the signals using aLIGO's Power Spectral Density (PSD) at the "Zero-detuned High Power" design sensitivity [85] shown in Figure 3. We have also ignored glitches, blips, and other transient sources of detector noise for now. This is in line with previous studies, which have first showcased a machine learning algorithm for LIGO data analysis using simulated noise [68, 72], and then followed up by an independent study where the algorithm is tested using real aLIGO noise [71]. Our analysis, using real aLIGO data, will be presented in a separate publication.

### Obtaining Data

Data from Simulations

Supervised deep learning algorithms are far more effective when trained with very large datasets. Obtaining high quality training data has been a difficult and cumbersome task in most applications of DNNs, such as object recognition in images, speech and text processing, etc. Fortunately, we do not face this issue, since we can take advantage of scientific simulations to produce the necessary data for training.

Source of GW waveform

Over the last decade, sophisticated techniques have been developed to perform accurate 3-dimensional NR simulations of merging BHs [84, 86] on HPC facilities. For the analysis at hand, we use Effective-One-Body (EOB) waveforms that describe GWs emitted by quasi-circular, non-spinning BBHs [38]. We extracted the final 1 second window of each template for our analysis.

Training & Testing

We have split the data into separate sets for training and testing. For the training dataset, the BBHs component masses are in the range $5M_\odot$ to $75M_\odot$ in steps of $1M_\odot$. The testing dataset has intermediate component masses, i.e., masses separated from values in the training dataset by $0.5M_\odot$. By not having overlapping values in the training and testing sets, one can ensure that the network is not overfitting, i.e., memorizing only the inputs shown to it without learning to generalize to new inputs. The distribution of component masses, and a template from the training and testing sets, is shown in Fig. 4. Subsequently, we shifted the location of the peak of each signal randomly within an interval of 0.2 seconds in both the training and testing sets to make the DNNs more robust with respect to time translations. Next, we superimposed different realizations of Gaussian white noise on top of the signals over multiple iterations, thus amplifying the size of the datasets. The power of the noise was adjusted according to the desired SNR for each training session. We then standardized the inputs to have zero mean and unit variance to make the training process easier [87].

The final training sets at each SNR were produced from

The final quantity

$\sim$ 2500 templates of GWs from BBH mergers by adding multiple batches of noise and shifting in time. It is also a standard practice to use a validation set to monitor the performance on unseen data during training in order to prevent overfitting. The validation and testing sets at each SNR were generated from a different set of $\sim$ 2500 templates by superimposing different noise realizations.

### Designing Neural Networks

这里迁移学习的手段不太好, 扩展性不好, 不是传统迁移学习的理解和操作手段.

Transfer Learning

We used very similar DNN architectures for both the classifier and predictor, which demonstrates the versatility of this method. The only difference was the addition of a softmax layer to the classifier to obtain probability estimates as the outputs. Our strategy was to first train the predictor on the datasets labeled with the BBH masses, and then transfer the weights of this pre-trained network to initialize the classifier and then train it on datasets with 50% random noise. This transfer learning process reduced the training time required for the classifier, while also slightly improving its accuracy at low SNR.

How is layers?

We designed simple DNNs from the ground up. Overall, we tested around 80 configurations of DNNs ranging from 1 to 4 convolutional layers and 1 to 3 fully connected layers (also called linear layers) similar to [88], but modified for time-series inputs. Among these, we discovered that a design for the classifier with 3 convolutional layers followed by 2 fully connected layers yielded good results with fastest inference speed for the datasets that we are considering. We tried adding a few recent developments such as batch normalization [89] and dropout [90] layers. However, we did not use them in our final design as they did not provide significant improvements for the simple problem we are considering. The addition of noise to the signals during the training process serves as a form of regularization in itself. Many of the layers have parameters, commonly known as hyperparameters, which we had to tune manually via a randomized trial-and-error procedure.

How is params. in layers?

Depth is a hyperparameter which determines the number of filters in each convolutional layer. Our choices for depth in the consecutive layers were 16, 32, and 64 respectively. We used kernel sizes of 16, 8, and 8 for the convolutional layers and 4 for all the (max) pooling layers. Stride, which specifies the shift between the receptive fields of adjacent neurons, was chosen to be 1 for all the convolution layers and 4 for all the pooling layers. Dilation determines the overall size of each receptive field, which could be larger than the kernel size by having gaps in between. Here, it is a measure of the temporal extend of the convolutions. We observed that using dilation of 4 in the final two convolution layers improved the performance. The final layout of our classifier DNN is shown in Fig. 5.

Deeper CNN?

Deeper networks are expected to provide further improvements in accuracy although at the cost of slower evaluation speed. To show this, we also designed a deeper net, shown in Fig. 6, with 4 convolution layers and 3 fully connected layers that had comparable sensitivity for detection and significantly

| | | |
|---|---|---|
| | Input | vector (size: 8192) |
| 1 | Reshape | matrix (size: 1 × 8192) |
| 2 | Convolution | matrix (size: 16 × 8177) |
| 3 | Pooling | matrix (size: 16 × 2044) |
| 4 | ReLU | matrix (size: 16 × 2044) |
| 5 | Convolution | matrix (size: 32 × 2016) |
| 6 | Pooling | matrix (size: 32 × 504) |
| 7 | ReLU | matrix (size: 32 × 504) |
| 8 | Convolution | matrix (size: 64 × 476) |
| 9 | Pooling | matrix (size: 64 × 119) |
| 10 | ReLU | matrix (size: 64 × 119) |
| 11 | Flatten | vector (size: 7616) |
| 12 | Linear Layer | vector (size: 64) |
| 13 | ReLU | vector (size: 64) |
| 14 | Linear Layer | vector (size: 2) |
| | Output | vector (size: 2) |

FIG. 5. **Architecture of deep neural network**. This is the deep dilated 1D CNN, modified to take time-series inputs, that we designed for prediction which outputs two real-valued numbers for the two component masses of the BBH system. For classification we simply added a softmax layer after the 14th layer to obtain the probabilities for two classes, i.e., "True" or "False". The input is the time-series sampled at 8192Hz and the output is either the probability of each class or the value of each parameter. Note that the number of neurons in layer 14 can be increased to add more categories for classification or more parameters for prediction. The size of this net is about 2*MB*.

better performance for parameter estimation. Although this design performed slightly better, it was a factor of 5 slower on a GPU for evaluation. This net had convolution layers having kernel sizes were 16, 16, 16, and 32 with dilations 1, 2, 2, and 2 respectively. The pooling layers all had kernel size 4 and stride 4.

How is Loss?

A loss function (cost function) is required to compute the error after each iteration by measuring how close the outputs are with respect to the target values. We designed a mean absolute relative error loss function for the predictor. For classification, we used the standard cross-entropy loss function.

### Training Strategy

Colored or not for Transfer Learning

We spent significant effort on hyperparameter optimization, to design architectures of the CNNs by trial and error. First, we used Gaussian white noise without whitening the signals i.e., a flat PSD, to determine the optimal architectures of the DNNs. We found that this design was also optimal for signals whitened with the Zero-Detuned PSD of aLIGO. This indicates that the same architecture will perform well on wide variety of PSDs. Once we chose the best performing DNNs, we trained it for about a total of 10 hours. We relied on the neural network functionality in the Wolfram Language, *Mathematica*, based internally on the open-source MXNet framework [91], which utilizes the CUDA deep learning library

| | | |
|---|---|---|
| | Input | vector (size: 8192) |
| 1 | Reshape | matrix (size: 1 × 8192) |
| 2 | Convolution | matrix (size: 64 × 8177) |
| 3 | Pooling | matrix (size: 64 × 2044) |
| 4 | ReLU | matrix (size: 64 × 2044) |
| 5 | Convolution | matrix (size: 128 × 2014) |
| 6 | Pooling | matrix (size: 128 × 503) |
| 7 | ReLU | matrix (size: 128 × 503) |
| 8 | Convolution | matrix (size: 256 × 473) |
| 9 | Pooling | matrix (size: 256 × 118) |
| 10 | ReLU | matrix (size: 256 × 118) |
| 11 | Convolution | matrix (size: 512 × 56) |
| 12 | Pooling | matrix (size: 512 × 14) |
| 13 | ReLU | matrix (size: 512 × 14) |
| 14 | Flatten | vector (size: 7168) |
| 15 | Linear Layer | vector (size: 128) |
| 16 | ReLU | vector (size: 128) |
| 17 | Linear Layer | vector (size: 64) |
| 18 | ReLU | vector (size: 64) |
| 19 | Linear Layer | vector (size: 2) |
| | Output | vector (size: 2) |

FIG. 6. **Architecture of deeper neural network**. This is the deeper version of the CNN, modified to take time-series inputs, that we designed for parameter estimation. The input is the time-series sampled at 8192Hz and the output is the predicted value of each parameter. This can be converted to a classifier by adding a softmax layer after layer 19 to obtain the probability for a detection. Note that the number of neurons in layer 19 can be increased to add more categories for classification or more parameters for prediction. The 2 neurons in the final layer outputs the 2 parameters corresponding to the individual masses of BBHs. The size of this net is approximately 23*MB*.

(cuDNN) [44] for acceleration with NVIDIA GPUs. We used the ADAM [63] method as our learning algorithm.

Transfer from high SNR for predictor.

During this process, we developed a new strategy to improve the performance and reduce training times of the DNNs. By starting off training the predictor on inputs having high SNR ($\geq$ 100) and then gradually increasing the noise in each subsequent training session until a final SNR distribution randomly sampled in the range 5 to 15, we observed that the performance can be quickly maximized for low SNR, while remaining accurate for signals with very high SNR. For instance, we obtained about 11% error when trained using this scheme with gradually decreasing SNR and only about 21% mean error at parameter estimation on the test set when directly trained on the same range of SNR (5-15). Furthermore, we found that the classifier performs significantly better (with an increase from 96% to 99% accuracy on one of our test sets) when its initial weights are transfered from the fully trained predictor, i.e., the classifier was created by simply adding a softmax layer to the trained predictor and then trained on the dataset of signals and noise. We expect these techniques would be useful for training neural networks, in general, with noisy data.

Transfer from predictor.

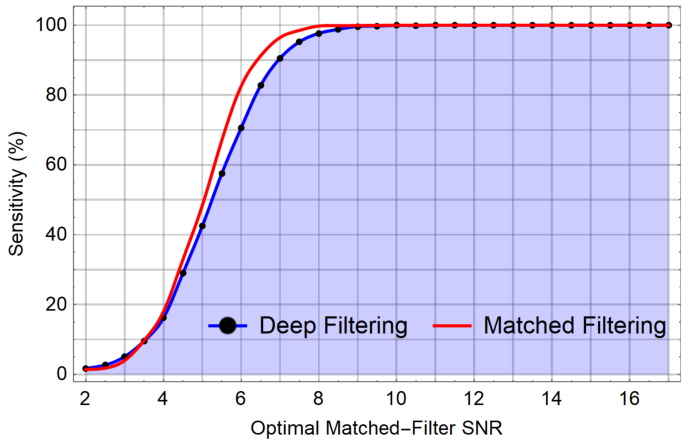


FIG. 7. **Sensitivity of detection with smaller net.** This is the sensitivity (fraction of signals detected) of the shallower classifier as a function of SNR on the test set. Note that the sensitivity was measured with the same classifier after training once over the entire range of SNR, i.e., without specifically re-training it for each SNR. This curve saturates at sensitivity of 100% for SNR $\geq$ 10, i.e, signals with SNR $\geq$ 10 are always detected. The single detector false alarm rate was tuned to be about 0.5% for this classifier. Note that the optimal matched-filter SNR is on average proportional to 12.9 $\pm$ 1.4 times the ratio of the amplitude of the signal to the standard deviation of the noise for our test set. This implies that Deep Filtering is capable of detecting signals significantly weaker than the background noise.

??

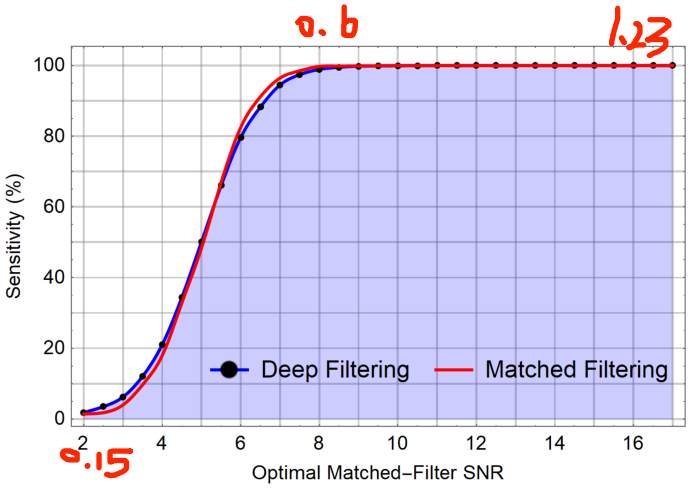


FIG. 8. **Sensitivity of detection with deeper net.** This is the sensitivity of the deeper classifier as a function of SNR on the test set. Note that this sensitivity was also measured with the same classifier after training once over the entire range of SNR, i.e., without specifically re-training it for each SNR. This curve saturates at sensitivity of 100% for SNR $\geq$ 9, i.e, signals with SNR $\geq$ 9 are always detected. The single detector false alarm rate was tuned to be approximately 0.5% for this classifier.

## IV. RESULTS

We trained our classifier to achieve 100% sensitivity for signals with SNR $\geq$ 10 and a single detector false alarm rate less than 0.6%. Note that the false alarm rate of Deep

Sensitivity with SNR
FAR

Filtering can be significantly decreased by combining classifications on multiple detector inputs and by computing the overlap of the template predicted by Deep Filtering with the input to confirm each detection. The sensitivity of this classifier as a function of SNR is shown in Fig. 7. The deeper classifier obtained slightly better sensitivity as shown in Fig. 8

Compare other MLs

For comparison, we trained standard implementations of all commonly used machine learning classifiers— Random Forest, Support Vector Machine, k-Nearest Neighbors, Hidden Markov Model, Shallow Neural Networks, Naive Bayes, and Logistic Regression — along with the DNNs on a simpler training set of 8000 elements for fixed total mass and peak signal amplitude. Unlike DNNs, none of these algorithms were able to directly handle raw noisy data even for this simple problem as shown in Fig. 12.

Predictor with large SNR

Our predictor was able to successfully measure the component masses given noisy GWs, that were not used for training, with an error of the same order as the spacing between templates for SNR $\geq$ 13. The deeper predictor consistently outperformed matched-filtering. At very large SNR, over 50, we could train both the predictors to have relative error less than 5%, whereas the error with matched-filtering using the same templates was always greater than 11% with the given template bank. This means that, unlike matched-filtering, our algorithm is able to automatically perform interpolation between the known templates to predict intermediate values. The variation in relative error against SNR for each architecture of the DNNs is shown in Fig. 9 and Fig. 10. The largest relative errors were concentrated at lower masses, because a small variation in predicted masses led to larger relative errors in this region.

Estimate error in parameter -space

We can estimate the distribution of errors and uncertainties *empirically* at each region of the parameter-space. We observed that the errors closely follow Gaussian normal distributions for each input for SNR ($\geq$ 9), allowing easier characterization of uncertainties. Once we obtain initial estimates for the parameters via Deep Filtering, traditional techniques may be rapidly applied using only a few templates near these predictions to cross-validate our detection and parameter estimates and to measure uncertainties. There are also emerging techniques to estimate quantify in the predictions of CNNs [92], which may be applied to this method.

ML predict bad for param.

After testing common machine learning techniques including Linear Regression, k-Nearest Neighbors, Shallow Neural Networks, Gaussian Process Regression, and Random Forest on the simpler problem with fixed total mass, we observed that, unlike DNNs, they could not predict even a single parameter (mass-ratio at fixed total mass) accurately, as evident from Fig. 12, when trained directly on time-series data.

Try to identify New GW

Having trained our DNNs to detect and characterize quasi-circular, non-spinning BBH signals, we assessed their capabilities to identify new classes of GW signals, beyond our original training and testing sets. We used two distinct types of signals that were *not* considered during the training stage, namely: (i) moderately eccentric NR simulations (approximate eccentricity of 0.1 when entering aLIGO band), that we recently generated with the open-source, Einstein Toolkit [84]

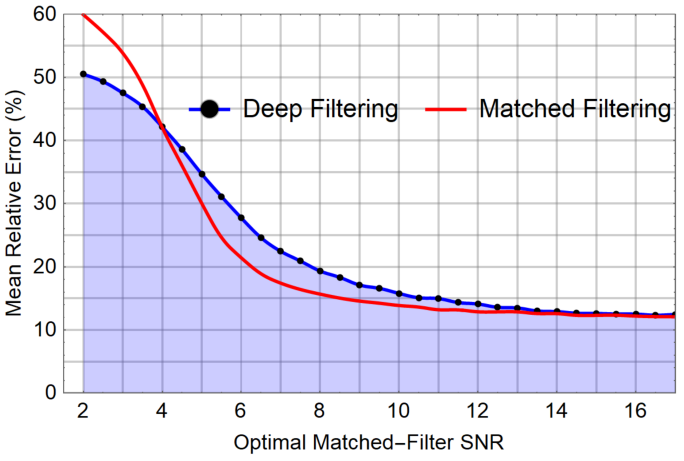


FIG. 9. **Error in parameter estimation with smaller net**. This shows the mean percentage error of estimated masses on our testing sets at each SNR using the predictor DNN with 3 convolution layers. The DNN was trained only once over the range of SNR and was then tested at different SNR, without re-training. Note that a mean relative error less than 20% was obtained for SNR $\geq$ 8 . At high SNR, the mean error saturates at around 11%. See Fig. 10 for the results with the deeper version of the predictor.

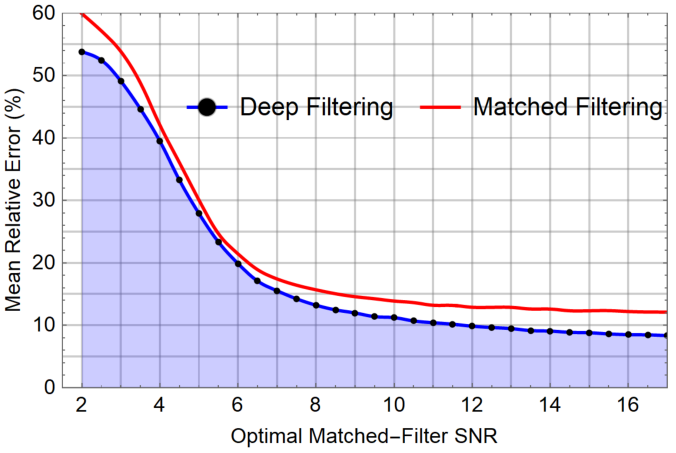


FIG. 10. **Error in parameter estimation with deeper net**. This shows the mean percentage error of estimated masses on our testing sets at each SNR using the deeper CNN with 4 convolution layers. Note that a mean relative error less than 15% was obtained for SNR $\geq$ 7 . At high SNR, the mean error saturates at around 7%. Note that we were able to optimize the predictor to have less than 3% error for very high SNR ($\geq$ 50), which demonstrates the ability of Deep Filtering to learn patterns connecting the templates and effectively interpolate to intermediate points in parameter space.

using the Blue Waters petascale supercomputer; and (ii) NR waveforms from the SXS catalog [93] that describe spin-precessing, quasi-circular BBHs—each BH having spin $\geq$ 0.5 oriented in random directions [93]. Sample waveforms of these GW classes as shown in Fig. 13. Since these NR simulations scale trivially with mass, we enlarged the data by rescaling the signals to have different total masses. Thereafter, we

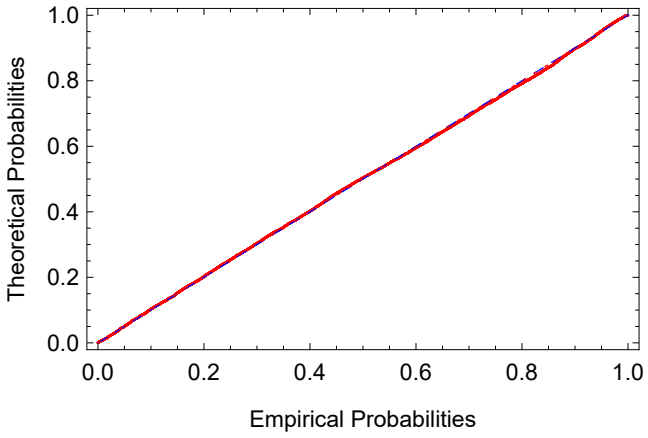


FIG. 11. **P-P plot of errors in parameter estimation** This is a P-P (probability) plot of the distribution of errors in predicting $m_1$ for test parameters $m_1 = 57M_\odot$ and $m_2 = 33M_\odot$, superimposed with different realizations of noise at SNR = 9. The best-fit is a Gaussian normal distribution with mean $= 1.5M_\odot$ and standard deviation $= 4.1M_\odot$. The errors have similar Gaussian distributions in other regions of the parameter-space as well.

whitened them and added different realizations of noise, in the same manner as before, to produce test sets.

We have found that both the classifiers detected all these signals with nearly the same rate as the original test set, with 100% sensitivity for SNR $\geq$ 10. Remarkably, the predictor quantified the component masses of our eccentric simulations for SNR $\geq$ 12 with a mean relative error less than 20% for mass-ratios $q = \{1, 2, 3, 4\}$, and less than 30% for $q = 5.5$ respectively. For the spin-precessing systems we tested, with SNR $\geq$ 12, the mean error in predicting the masses was less than 20% for $q = \{1, 3\}$, respectively.

These findings are very encouraging, since recent analyses have made evident that existing aLIGO algorithms are not capable of accurately detecting or reconstructing the parameters of eccentric signals [94–96], and do not cover spin-precessing systems [29]. This ability to generalize to new categories of signals, without being shown any examples, means that DNN-based pipelines can increase the depth of existing GW detection algorithms without incurring in any additional computational expense.

Speed when training.

Furthermore, our simple classifier and predictor are only *2MB* in size each, yet they achieve excellent results. The average time taken for evaluating them per input of 1 second duration is approximately 6.7 milliseconds, and 106 microseconds using a single CPU and GPU respectively. The deeper predictor net, which is about *23MB*, achieves slightly better accuracy at parameter estimation but takes about 85 milliseconds for evaluation on the CPU and 535 microseconds on the GPU, which is still orders of magnitude faster than real-time. Note that the current deep learning frameworks are not well optimized for CPU evaluation. For comparison, we estimated an evaluation time of 1.1 seconds for time-domain matched-filtering on the same CPU (using 2-cores) with the same template bank of clean signals used for training, the results are shown in Fig. 14. This extremely fast inference rate indicates that real-time analysis can be carried out with a single CPU or GPU, even with DNNs that are significantly larger and trained over a much larger template banks of millions of signals. For example, a state-of-the-art CNN for image recognition [97, 98] has hundreds of layers (61*MB* in size) and is trained with over millions of examples to recognize thousands of different categories of objects. This CNN can process significantly larger inputs, each having dimensions $224 \times 224 \times 3$, using a single GPU with a mean time of 6.5 milliseconds per input. Note that these CNNs can be trained on millions of inputs in a few hours using parallel GPUs [99].

How multi-detector ?

For applying the `Deep Filtering` method to a multi-detector scenario, we simply need to apply our nets pre-trained for single detector inference separately to each detector and check for coincident detections with similar parameter estimates. Enforcing coincident detections would decrease our false alarm probability, from about 0.59% to about 0.003%. Once the `Deep Filtering` pipeline detects a signal then traditional matched-filtering may be applied with a select few templates around the estimated parameters to cross-validate the event and estimate confidence measure. Since only a few templates need to be used with this strategy, existing challenges to extend matched-filtering for higher dimensional GW searches may thus be overcome, allowing real-time analysis with minimal computational resources.

## V. DISCUSSION

The results we obtained with our prototype DNNs exceeded our expectations with high detection rate and low prediction errors even for signals with very low SNR. Initially, we had trained a DNN to predict only the mass-ratios at a fixed total mass. Extending this to predict two component masses was as simple as adding an extra neuron to the output layer, which suggests that it would be straightforward to extend our method to predict any number of parameters such as spins, eccentricities, etc. By incorporating examples of transient detector noise in the training set, the DNNs can also be taught to automatically ignore or classify glitches. We have only explored very simple DNNs in this first study, therefore, it is expected that more complex DNNs would improve the accuracy of interpolation between GW templates for prediction as well as the sensitivity at low SNR, while retaining real-time performance.

Based on our preliminary results, we expect `Deep Filtering` to be able to learn from and adapt to the characteristics of LIGO noise when trained with real data. The performance of this algorithm with real aLIGO data, especially in the presence of glitches and for the detection of true GW events, will be demonstrated in a following work.

Deep learning is known to be highly scalable, overcoming what is known as the curse of dimensionality [100]. This intrinsic ability of DNNs to take advantage of large datasets is a unique feature to enable simultaneous GW searches over a higher dimensional parameter-space that is beyond the reach of existing algorithms. Furthermore, DNNs are excellent at

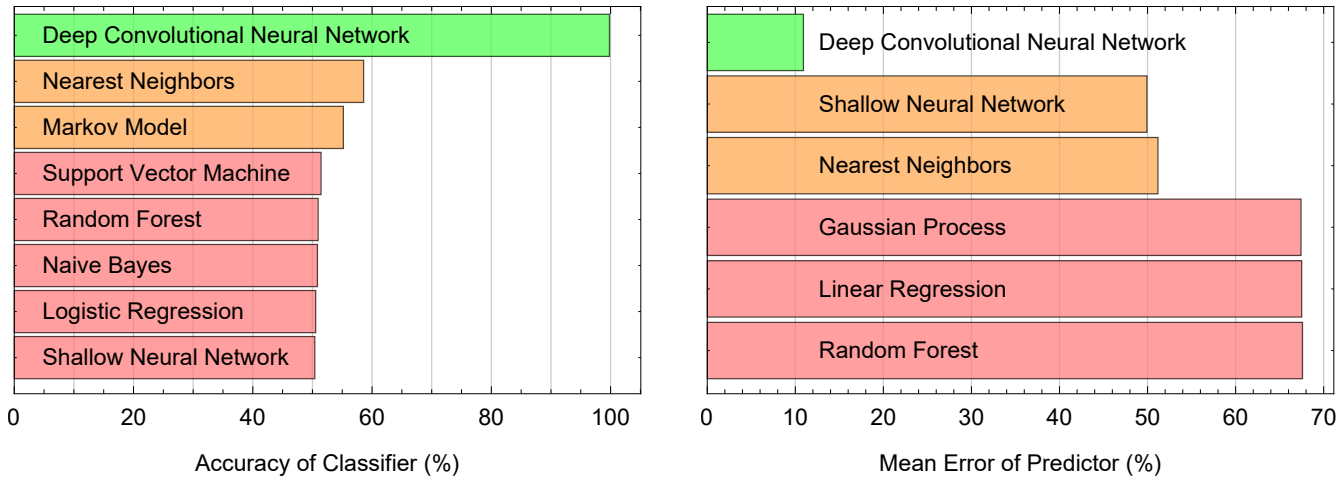


FIG. 12. **Comparison with other methods.** Left panel: This is the accuracy of different machine learning methods for detection after training each with roughly 8000 elements, half of which contained noisy signals with a fixed peak power, less than the background noise, and constant total mass, with the other half being pure noise with unit standard deviation.An accuracy of 50% can be obtained by randomly guessing. Right panel: This is the mean relative error obtained by various machine learning algorithms for predicting a single parameter, i.e., mass-ratio, using a training set containing about 8000 signals with fixed amplitude = 0.6 added to white noise with unit standard deviation. Note that scaling these methods to predict multiple parameters is often difficult, whereas it simply involves adding neurons to the final layer of neural networks.

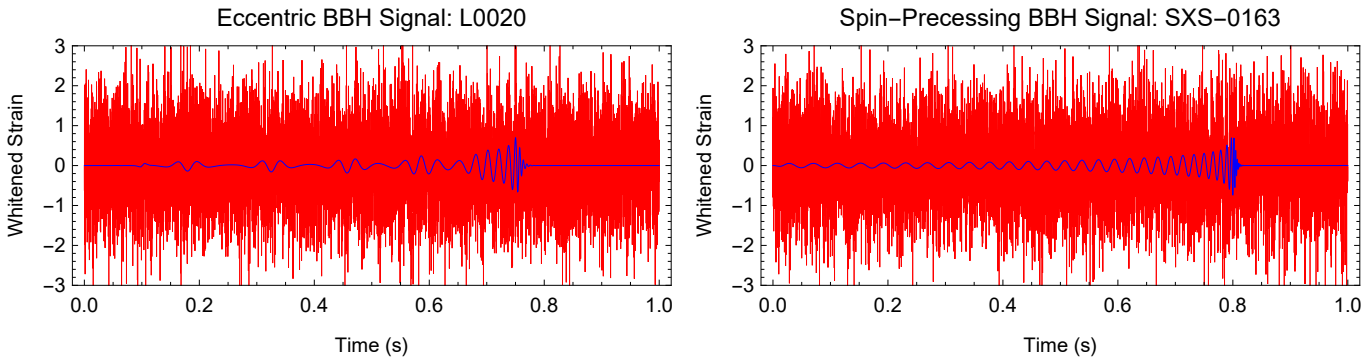


FIG. 13. **New types of signals**. Left panel: This waveform was obtained from one of our NR simulations of eccentric BBH merger that has mass-ratio 5.5, total mass about $90M_\odot$, and an initial eccentricity $e_0 = 0.2$ when it enters the aLIGO band. Our `Deep Filtering` pipeline successfully detected this signal, even when the total mass was scaled between $50M_\odot$ and $90M_\odot$, with 100% sensitivity (for SNR $\geq$ 10) and predicted the component masses with a mean relative error $\leq 30\%$ for SNR $\geq$ 12. Right panel: One of the spin-precessing waveforms obtained from the NR simulations in the SXS catalog with component masses equal to $25M_\odot$ each. The individual spins are each 0.6 and oriented in un-aligned directions. Our DNNs also successfully detected this signal, even when the total mass was scaled between $40M_\odot$ and $100M_\odot$, with 100% sensitivity for SNR $\geq$ 10 and predicted the component masses with a mean relative error $\leq 20\%$ for SNR $\geq$ 12.

generalizing or extrapolating to new data. We have shown that our DNNs, trained with only signals from non-spinning BHs on quasi-circular orbits, can detect and reconstruct the parameters of eccentric and spin-precessing compact sources that may go unnoticed with existing aLIGO detection algorithms [94–96, 101]. It is probable that our classifier is already capable of detecting even more types of signals, beyond what we have tested.

As our understanding of scientific phenomena improves and catalogs of NR simulations become available, new categories of detected and simulated GW sources can be easily added to the training datasets with minimal modifications to the architecture of DNNs. Multi-task learning [102] allows a single DNN to classify inputs into categories and sub-categories, while also performing parameter estimation for each type of signal. This means that simultaneous real-time searches for compact binary coalescence, GW bursts, supernovae, and other exotic events as well as classification of noise transients can be carried out under a single unified pipeline.

Our DNN algorithm requires minimal pre-processing. In principle, aLIGO's colored noise can be superimposed into the training set of GW templates, along with observed glitches. It has been recently found that deep CNNs are capable of automatically learning to perform band-pass filtering on raw time-series inputs [103], and that they are excellent at suppressing highly non-stationary colored noise [104] especially when incorporating real-time noise characteristics [105]. This suggests that manually devised pre-processing and whitening

Recent works for CNNs

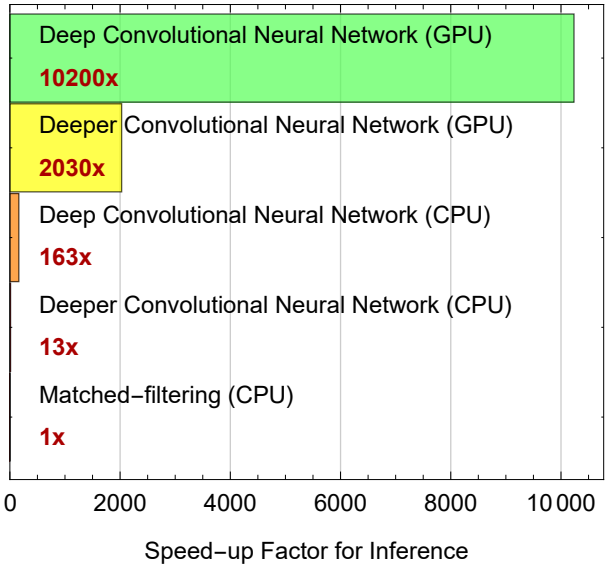


FIG. 14. **Speed-up of analysis**. The DNN-based pipeline is many orders of magnitude faster compared to matched-filtering (cross-correlation or convolution) against the same template bank of waveforms (tested on batches of inputs using both cores of an Intel Core i7-6500U CPU and an inexpensive NVIDIA GeForce GTX 1080 GPU for a fairer comparison). Note that the evaluation time of a DNN is constant regardless of the size of training data, whereas the time taken for matched-filtering is proportional to the number of templates being considered, i.e., exponentially proportional to the number of parameters. Therefore, the speed-up of `Deep Filtering` would be higher in practice, especially when considering larger template banks over a higher dimensional parameter space.

steps may be eliminated and raw aLIGO data can be fed to DNNs. This would be particularly advantageous since it is known that Fourier transforms are the bottlenecks of aLIGO pipelines [29].

Powerful modern hardware, such as GPUs, ASICs, or FPGAs, are essential to efficiently train DNNs. An ideal choice would be the new NVIDIA DGX-1 supercomputers dedicated for deep learning analytics located on-site at each of the LIGO labs. However, once DNNs are trained with a given aLIGO PSD, they can be more quickly re-trained, via transfer learning, during a detection campaign for recalibration in real-time based on the latest characteristics of each detectors' noise. Deep learning methods can also be immediately applied through distributed computing via citizen science campaigns such as Einstein@Home [106] as several open-source deep learning libraries, including MXNet, allow scalable distributed training and evaluation of neural networks simultaneously on heterogeneous devices, including smartphones and tablets. Low-power devices such as FPGAs and GPU chips dedicated for deep learning inference [46, 107, 108] may even be placed on the GW detectors to reduce data transfer issues and latency in analysis.

DNNs automatically extract and compress information by finding patterns within the training data, creating a dimensionally reduced model [109]. Our fully trained DNNs are each only 2*MB* (or 23*MB* for the deeper model) in size yet encodes all the relevant information from about 2500 GW templates (about 200*MB*, before the addition of noise) used to generate the training data. Once trained, analyzing a second of data takes only milliseconds with a single CPU and microseconds with a GPU. This means that real-time GW searches could be carried out by anyone with an average laptop computer or even a smartphone, while big datasets can be processed rapidly in bulk with inexpensive hardware and software optimized for inference. The speed, power efficiency, and portability of DNNs would allow rapidly analyzing the continuous stream of data from GW detectors or other astronomical facilities.

### Coincident Detection of GWs and EM Counterparts

BNS inspirals have been confirmed as the engines of short gamma ray bursts (sGRBs) [19, 22–26, 110–112]. We expect that future detections of NSBH mergers may confirm whether these systems are also the progenitors of sGRBs, and whether rapidly rotating hypernovae are the progenitors of long duration GRBs, collapsars, etc. [7, 27]. DNNs are particularly suited for image and video processing, therefore, they can be trained to simultaneously search for GW transients and their EM counterparts using telescopes' raw image data [113]. If the identification of an EM transient can be carried out quickly, we can interface this information with a DNN-based GW detection pipeline and vice-versa. Joint analyses of this nature will enable real-time multimessenger astrophysics searches.

Recent work suggests that space-based GW detectors such as the evolved Laser Interferometer Space Antenna (eLISA) [114, 115] will be able to detect stellar mass BBH systems weeks before they merge in the frequency band of ground-based GW detectors [116]. DNNs can be used to detect these sources in the eLISA and aLIGO frequency bands using a unified pipeline (on-board analysis may be possible in space with extremely power-efficient chips dedicated for deep learning inference). Furthermore, by training similar DNNs, low-latency classification algorithms to search for EM transients in the anticipated sky region where these events are expected to occur.

In summary, the flexibility and computational efficiency of DNNs could promote them as standard tools for multimessenger astrophysics.

### Scope for Improvements

One may construct a multi-dimensional template bank using available semi-analytical waveform models, and all available NR waveforms. Thereafter, one can superimpose samples of real aLIGO noise, and non-Gaussian noise transients, on these templates, and carry out an intensive training procedure with coincident time-series inputs from multiple detectors. Once this process is finished, the DNN may be used for real-time classification and parameter estimation, including sky localization, while being periodically re-trained with

more gravitational waveforms and recent aLIGO noise. Time-series inputs from multiple detectors may be provided directly to the CNNs and more neurons may be added in the final layer to predict more parameters such as spins, eccentricity, time difference, location in the sky, etc. The hyperparameters of the neural networks may be tuned, and more layers may be added to further improve the performance of `Deep Filtering`.

CNNs are limited by the fact that they can only use fixed length tensors as inputs and outputs and thus require a sliding window technique in practice. On the other hand, RNNs, the deepest of all neural networks, have cyclic internal structures and are well-suited for time-series analysis since they can make decisions based on a continuous stream of inputs rather than a vector of fixed length [50], however, they are harder to train [117]. A powerful type of RNN called LSTM (Long-Short-Term-Memory) [118] is capable of remembering long-term dependencies in the input sequence. Therefore RNNs [50] are ideal for processing temporal data as they can take inputs of variable lengths and have been remarkably successful at voice recognition problems [119]. We are developing sequence-to-sequence models with LSTM RNNs and CNNs which can be used to denoise the input time-series and produce the clean signal as output. This pre-processed data can then be fed into our `Deep Filtering` pipeline so as to further improve the sensitivity at very low SNR.

Stacking time-series datasets to produce multi-dimensional tensors can facilitate processing massive quantities of data efficiently on modern hardware, for e.g., to find signals that are very long in duration like BNS inspirals. The accuracy of the DNNs can be further enhanced by training an ensemble of different models and averaging the results for each input [49].

aLIGO uses a variety of independent sensors to monitor the environment and assess data quality. Many algorithms are currently used to estimate periods which must be vetoed due to disturbances that lead to a loss in detector sensitivity. Data quality information from these auxiliary channels may also be incorporated to improve robustness of signal detection and parameter estimation in the presence of glitches and for detector characterization [120].

In a broader context, our results indicate that, given models or template banks of expected signals, `Deep Filtering` can be used as a generic tool for efficiently detecting and extracting highly noisy time-domain signals in any discipline.

## VI. CONCLUSION

We have presented a novel framework for signal processing that is tailored to enable real-time multimessenger astrophysics, and which can enhance existing data analysis techniques in terms of both performance and scalability. We exposed CNNs to time-series template banks of GWs, and allowed it to develop its own strategies to extract a variety of GW signals from highly noisy data. The DNN-based prototype introduced in this article provides a strong incentive to conduct a more comprehensive investigation and optimization of DNNs to build a new data analysis pipeline based on `Deep Filtering`, trained with real detector noise, including glitches, and the largest available template banks covering the entire parameter-space of signals, to incorporate glitch classification and to accelerate and broaden the scope of GW searches with aLIGO and future GW missions. We are currently collaborating with the developers of the `PyCBC` pipeline [29], which is routinely used for GW detection both in off-line and on-line mode, to implement `Deep Filtering` as a module to increase the science reach of GW astronomy.

The known scalability of deep learning to high-dimensional data allows the use of as many GW templates as needed to train DNNs to simultaneously target a broad class of astrophysically motivated GWs sources. More neurons may be added to encode as much astrophysical information as needed for predicting any number of parameters, and multi-task learning can unify detection and classification of different types of sources and glitches, as well as parameter estimation, with a single DNN. Therefore, we expect this approach will increase the depth and speed of existing GW algorithm allowing real-time online searches after being trained with template banks of millions or billions of waveforms.

The DNN-based pipeline can be used to provide instant alerts with accurate parameters for EM follow-up campaigns, and also to accelerate matched-filtering and detailed Bayesian parameter estimation methods. Each prediction made by the DNNs can be quickly verified by performing traditional template matching with only the templates close to the predicted parameters. While aLIGO matched-filtering pipelines do not cover GWs from spin-precessing and eccentric BBH mergers, we have shown that DNNs were able to automatically generalize well to these signals, even without using these templates for training, having similar detection rates for all signals and small errors in estimating parameters of low mass-ratio systems. We expect that including examples of all classes of known GW signals and noise transients while training would improve the performance across the entire range of signals. We are now working on including millions of spin-precessing and eccentric templates and developing methods to train on large-scale parallel GPU clusters.

Employing DNNs for multimessenger astrophysics offers unprecedented opportunities to harness hyper-scale AI computing with emerging hardware architectures, and cutting-edge software. In addition, the use of future exascale supercomputing facilities will be critical for performing improved HPC simulations that faithfully encode the gravitational and EM signatures of more types of sources, which will be used to *teach* these intelligent algorithms. We expect that our new approach will percolate in the scientific community and serve as a key step in enabling real-time multimessenger observations by providing immediate alerts for follow-up after GW events. As deep CNNs excel at image processing, applying the same technique to analyze raw telescope data may accelerate the subsequent search for transient EM counterparts. We also anticipate that our new methodology for processing signals hidden in noisy data will be useful in many other areas of engineering, science, and technology. Therefore, this work is laying the foundations to integrate diverse domains of expertise to enable and accelerate scientific discovery.

## ACKNOWLEDGMENTS

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. The eccentric numerical relativity simulations used in this article were generated on Blue Waters with the open source, community software, the Einstein Toolkit. We express our gratitude to Gabrielle Allen, Ed Seidel, Roland Haas, Miguel Holgado, Haris Markakis, Justin Schive, Zhizhen Zhao, other members of the NCSA Gravity Group, and Prannoy Mupparaju for their comments and interactions and to the many others who provided feedback on our manuscript. We thank Vlad Kindratenko for granting us unrestricted access to numerous GPUs and HPC resources in the Innovative Systems Lab at NCSA. We are grateful to NVIDIA for their generous donation of several Tesla P100 GPUs, which we used in our analysis. We also acknowledge Wolfram Research for technical assistance and for developing the software stack used to carry out this study and draft this publication.



[1] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al., Physical Review Letters **116**, 061102 (2016), arXiv:1602.03837 [gr-qc].

[2] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al., Physical Review Letters **116**, 241103 (2016), arXiv:1606.04855 [gr-qc].

[3] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al., Physical Review X **6**, 041015 (2016), arXiv:1606.04856 [gr-qc].

[4] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, *et al.*, Physical Review Letters **118**, 221101 (2017).

[5] F. Acernese *et al.*, Classical and Quantum Gravity **32**, 024001 (2015), arXiv:1408.3978 [gr-qc].

[6] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al., *Phys. Rev.* D **94**, 064035 (2016), arXiv:1606.01262 [gr-qc].

[7] C. D. Ott, Classical and Quantum Gravity **26**, 063001 (2009), arXiv:0809.0695.

[8] P. Mösta, B. C. Mundim, J. A. Faber, R. Haas, S. C. Noble, T. Bode, F. Löffler, C. D. Ott, C. Reisswig, and E. Schnetter, Classical and Quantum Gravity **31**, 015005 (2014), arXiv:1304.5544 [gr-qc].

[9] R. Haas, C. D. Ott, B. Szilagyi, J. D. Kaplan, J. Lippuner, M. A. Scheel, K. Barkett, C. D. Muhlberger, T. Dietrich, M. D. Duez, F. Foucart, H. P. Pfeiffer, L. E. Kidder, and S. A. Teukolsky, *Phys. Rev.* D **93**, 124062 (2016), arXiv:1604.00782 [gr-qc].

[10] E. Abdikamalov, S. Gossan, A. M. DeMaio, and C. D. Ott, *Phys. Rev.* D **90**, 044001 (2014), arXiv:1311.3678 [astro-ph.SR].

[11] L. E. Kidder, S. E. Field, F. Foucart, E. Schnetter, S. A. Teukolsky, A. Bohn, N. Deppe, P. Diener, F. Hébert, J. Lippuner, J. Miller, C. D. Ott, M. A. Scheel, and T. Vincent, Journal of Computational Physics **335**, 84 (2017), arXiv:1609.00098 [astro-ph.HE].

[12] S. Nissanke, M. Kasliwal, and A. Georgieva, *Astrophys. J.* **767**, 124 (2013), arXiv:1210.6362 [astro-ph.HE].

[13] Dark Energy Survey Collaboration, *MNRAS* **460**, 1270 (2016), arXiv:1601.00329.

[14] A. A. Abdo, M. Ajello, A. Allafort, L. Baldini, J. Ballet, G. Barbiellini, M. G. Baring, D. Bastieri, A. Belfiore, R. Bellazzini, and et al., *ApJS* **208**, 17 (2013), arXiv:1305.4385 [astro-ph.HE].

[15] J. A. Tyson, in *Survey and Other Telescope Technologies and Discoveries*, Proceedings of SPIE, Vol. 4836, edited by J. A. Tyson and S. Wolff (2002) pp. 10–20, astro-ph/0302102.

[16] L. Amendola, S. Appleby, D. Bacon, T. Baker, M. Baldi, N. Bartolo, A. Blanchard, C. Bonvin, S. Borgani, E. Branchini, C. Burrage, S. Camera, C. Carbone, L. Casarini, M. Cropper, C. de Rham, C. Di Porto, A. Ealet, P. G. Ferreira, F. Finelli, J. García-Bellido, T. Giannantonio, L. Guzzo, A. Heavens, L. Heisenberg, C. Heymans, H. Hoekstra, L. Hollenstein, R. Holmes, O. Horst, K. Jahnke, T. D. Kitching, T. Koivisto, M. Kunz, G. La Vacca, M. March, E. Majerotto, K. Markovic, D. Marsh, F. Marulli, R. Massey, Y. Mellier, D. F. Mota, N. Nunes, W. Percival, V. Pettorino, C. Porciani, C. Quercellini, J. Read, M. Rinaldi, D. Sapone, R. Scaramella, C. Skordis, F. Simpson, A. Taylor, S. Thomas, R. Trotta, L. Verde, F. Vernizzi, A. Vollmer, Y. Wang, J. Weller, and T. Zlosnik, Living Reviews in Relativity **16** (2013), 10.12942/lrr-2013-6, arXiv:1206.1225.

[17] N. Gehrels, D. Spergel, and WFIRST SDT Project, in *Journal of Physics Conference Series*, Journal of Physics Conference Series, Vol. 610 (2015) p. 012007, arXiv:1411.0313 [astro-ph.IM].

[18] ANTARES Collaboration, IceCube Collaboration, LIGO Scientific Collaboration, Virgo Collaboration, S. Adrián-Martínez, A. Albert, M. André, G. Anton, M. Ardid, J.-J. Aubert, and et al., ArXiv e-prints (2016), arXiv:1602.05411 [astro-ph.HE].

[19] The LIGO Scientific Collaboration and The Virgo Collaboration, ArXiv e-prints (2017), arXiv:1710.05832 [gr-qc].

[20] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al., Living Reviews in Relativity **19** (2016), 10.1007/lrr-2016-1, arXiv:1304.0670 [gr-qc].

[21] L. P. Singer, L. R. Price, B. Farr, A. L. Urban, C. Pankow, S. Vitale, J. Veitch, W. M. Farr, C. Hanna, K. Cannon, T. Downes, P. Graff, C.-J. Haster, I. Mandel, T. Sidery, and A. Vecchio, *Astrophys. J.* **795**, 105 (2014), arXiv:1404.5623 [astro-ph.HE].

[22] LIGO Scientific Collaboration, Virgo Collaboration, F. GBM, INTEGRAL, IceCube Collaboration, AstroSat Cadmium Zinc Telluride Imager Team, IPN Collaboration, The Insight-Hxmt Collaboration, ANTARES Collaboration, The Swift Collaboration, AGILE Team, The 1M2H Team, The Dark Energy Camera GW-EM Collaboration, the DES Collaboration, The

DLT40 Collaboration, GRAWITA, :, GRAvitational Wave Inaf TeAm, The Fermi Large Area Telescope Collaboration, ATCA, :, A. Telescope Compact Array, ASKAP, :, A. SKA Pathfinder, Las Cumbres Observatory Group, OzGrav, DWF, AST3, CAASTRO Collaborations, The VINROUGE Collaboration, MASTER Collaboration, J-GEM, GROWTH, JAGWAR, C. NRAO, TTU-NRAO, NuSTAR Collaborations, Pan-STARRS, The MAXI Team, T. Consortium, KU Collaboration, N. Optical Telescope, ePESSTO, GROND, T. Tech University, SALT Group, TOROS, :, Transient Robotic Observatory of the South Collaboration, The BOOTES Collaboration, MWA, :, M. Widefield Array, The CALET Collaboration, IKI-GW Follow-up Collaboration, H. E. S. S. Collaboration, LOFAR Collaboration, LWA, :, L. Wavelength Array, HAWC Collaboration, The Pierre Auger Collaboration, ALMA Collaboration, Euro VLBI Team, Pi of the Sky Collaboration, The Chandra Team at McGill University, DFN, :, D. Fireball Network, ATLAS, H. Time Resolution Universe Survey, RIMAS, RATIR, and S. South Africa/MeerKAT, ArXiv e-prints (2017), arXiv:1710.05833 [astro-ph.HE].

[23] D. Eichler, M. Livio, T. Piran, and D. N. Schramm, Nature **340**, 126 (1989).

[24] B. Paczynski, *Astrophys. J.* Lett **308**, L43 (1986).

[25] R. Narayan, B. Paczynski, and T. Piran, *Astrophys. J.* Lett **395**, L83 (1992), astro-ph/9204001.

[26] C. S. Kochanek and T. Piran, Astrophysical Journal **417**, L17 (1993), arXiv:astro-ph/9305015 [astro-ph].

[27] E. S. Phinney, in *The Astronomy and Astrophysics Decadal Survey*, Astronomy, Vol. 2010 (2009) arXiv:0903.0098 [astro-ph.CO].

[28] T. B. Littenberg, B. Farr, S. Coughlin, and V. Kalogera, *Astrophys. J.* **820**, 7 (2016), arXiv:1601.02661 [astro-ph.HE].

[29] S. A. Usman, A. H. Nitz, I. W. Harry, C. M. Biwer, D. A. Brown, M. Cabero, C. D. Capano, T. Dal Canton, T. Dent, S. Fairhurst, M. S. Kehl, D. Keppel, B. Krishnan, A. Lenon, A. Lundgren, A. B. Nielsen, L. P. Pekowsky, H. P. Pfeiffer, P. R. Saulson, M. West, and J. L. Willis, Classical and Quantum Gravity **33**, 215004 (2016), arXiv:1508.02357 [gr-qc].

[30] K. Cannon, R. Cariou, A. Chapman, M. Crispin-Ortuzar, N. Fotopoulos, M. Frei, C. Hanna, E. Kara, D. Keppel, L. Liao, S. Privitera, A. Searle, L. Singer, and A. Weinstein, *Astrophys. J.* **748**, 136 (2012), arXiv:1107.2665 [astro-ph.IM].

[31] B. P. Abbott, R. Abbott, T. D. Abbott, M. R. Abernathy, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, and et al., *Phys. Rev.* D **93**, 122004 (2016), arXiv:1602.03843 [gr-qc].

[32] N. J. Cornish and T. B. Littenberg, Classical and Quantum Gravity **32**, 135012 (2015), arXiv:1410.3835 [gr-qc].

[33] R. Smith, S. E. Field, K. Blackburn, C.-J. Haster, M. Pürrer, V. Raymond, and P. Schmidt, *Phys. Rev.* D **94**, 044031 (2016), arXiv:1604.08253 [gr-qc].

[34] I. Harry, S. Privitera, A. Bohé, and A. Buonanno, *Phys. Rev.* D **94**, 024012 (2016), arXiv:1603.02444 [gr-qc].

[35] Y. Lecun, Y. Bengio, and G. Hinton, Nature **521**, 436 (2015).

[36] Y. LeCun and Y. Bengio, in *The Handbook of Brain Theory and Neural Networks*, edited by M. A. Arbib (MIT Press, 1998) Chap. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258.

[37] K. Belczynski, D. E. Holz, T. Bulik, and R. O'Shaughnessy, Nature **534**, 512 (2016), arXiv:1602.04531 [astro-ph.HE].

[38] A. Taracchini, A. Buonanno, Y. Pan, T. Hinderer, M. Boyle, D. A. Hemberger, L. E. Kidder, G. Lovelace, A. H. Mroué, H. P. Pfeiffer, M. A. Scheel, B. Szilágyi, N. W. Taylor, and A. Zenginoglu, *Phys. Rev.* D **89**, 061502 (2014), arXiv:1311.2544 [gr-qc].

[39] D. J. C. Mackay, *Information Theory, Inference and Learning Algorithms, by David J. C. MacKay, pp. 640. ISBN 0521642981. Cambridge, UK: Cambridge University Press, October 2003.* (2003) p. 640.

[40] C. J. Moore, C. P. L. Berry, A. J. K. Chua, and J. R. Gair, *Phys. Rev.* D **93**, 064001 (2016), arXiv:1509.04066 [gr-qc].

[41] C. J. Moore and J. R. Gair, Physical Review Letters **113**, 251101 (2014), arXiv:1412.3657 [gr-qc].

[42] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. Katsaggelos, S. Larson, T. K. Lee, C. Lintott, T. Littenberg, A. Lundgren, C. Oesterlund, J. Smith, L. Trouille, and V. Kalogera, ArXiv e-prints (2016), arXiv:1611.04596 [gr-qc].

[43] D. George, H. Shen, and E. A. Huerta, ArXiv e-prints (2017), arXiv:1706.07446 [gr-qc].

[44] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, CoRR **abs/1410.0759** (2014).

[45] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16 (USENIX Association, 2016) pp. 265–283.

[46] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15 (ACM, 2015) pp. 161–170.

[47] T. E. Potok, C. D. Schuman, S. R. Young, R. M. Patton, F. Spedalieri, J. Liu, K.-T. Yao, G. Rose, and G. Chakma, in *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*, MLHPC '16 (IEEE Press, 2016) pp. 47–55.

[48] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, Science **345**, 668 (2014), http://science.sciencemag.org/content/345/6197/668.full.pdf.

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

[50] J. Schmidhuber, Neural Networks **61**, 85 (2015).

[51] Y. Bengio, A. Courville, and P. Vincent, IEEE Trans. Pattern Anal. Mach. Intell. **35**, 1798 (2013).

[52] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, ArXiv e-prints (2016), arXiv:1609.03499 [cs.SD].

[53] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, Journal of Big Data **2**, 1 (2015).

[54] M. Nielsen, *Neural Networks and Deep Learning* (2016) e-book.

[55] D. Graupe, *Principles of Artificial Neural Networks, 3rd edition, pp. 500, ISBN 9789814522755. University of Illinois, Chicago, USA. World Scientific* (2013).

[56] F. Rosenblatt, Psychological Review **65**, 386 (1958).

[57] M. Minsky and S. Papert, "Perceptrons : an introduction to computational geometry," (1969).

[58] K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**, 359 (1989).

[59] K. Jarrett, K. Kavukcuoglu, and Y. Lecun, "What is the best multi-stage architecture for object recognition?".

[60] "Wikimedia Commons: Artifical Neural Network," https://upload.wikimedia.org/wikipedia/commons/thumb/e/e4/Artificial_neural_network.svg/2000px-Artificial_neural_network.svg.png, accessed: 12-30-2016.

[61] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, in *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop* (Springer-Verlag, 1998) pp. 9–50.

[62] S. Ruder, CoRR **abs/1609.04747** (2016).

[63] D. P. Kingma and J. Ba, CoRR **abs/1412.6980** (2014).

[64] K. Fukushima, Biological Cybernetics **36**, 193 (1980).

[65] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Advances in Neural Information Processing Systems 25* (Curran Associates, Inc., 2012) pp. 1097–1105.

[66] F. Yu and V. Koltun, in *ICLR* (2016).

[67] B. J. Owen and B. S. Sathyaprakash, *Phys. Rev.* D **60**, 022002 (1999), gr-qc/9808076.

[68] J. Veitch, V. Raymond, B. Farr, W. Farr, P. Graff, S. Vitale, B. Aylott, K. Blackburn, N. Christensen, M. Coughlin, W. Del Pozzo, F. Feroz, J. Gair, C.-J. Haster, V. Kalogera, T. Littenberg, I. Mandel, R. O'Shaughnessy, M. Pitkin, C. Rodriguez, C. Röver, T. Sidery, R. Smith, M. Van Der Sluys, A. Vecchio, W. Vousden, and L. Wade, *Phys. Rev.* D **91**, 042003 (2015), arXiv:1409.7215 [gr-qc].

[69] P. Graff, F. Feroz, M. P. Hobson, and A. Lasenby, *MNRAS* **421**, 169 (2012), arXiv:1110.2997 [astro-ph.IM].

[70] N. Mukund, S. Abraham, S. Kandhasamy, S. Mitra, and N. S. Philip, Phys. Rev. D **95**, 104059 (2017).

[71] J. Powell *et al.*, Classical and Quantum Gravity **34**, 034002 (2017), arXiv:1609.06262 [astro-ph.IM].

[72] J. Powell, D. Trifirò, E. Cuoco, I. S. Heng, and M. Cavaglià, Classical and Quantum Gravity **32**, 215012 (2015), arXiv:1505.01299 [astro-ph.IM].

[73] D. George, H. Shen, and E. A. Huerta, ArXiv e-prints (2017), arXiv:1706.07446 [gr-qc].

[74] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. Katsaggelos, S. Larson, T. K. Lee, C. Lintott, T. Littenberg, A. Lundgren, C. Oesterlund, J. Smith, L. Trouille, and V. Kalogera, ArXiv e-prints (2016), arXiv:1611.04596 [gr-qc].

[75] S. Bahaadini, N. Rohani, S. Coughlin, M. Zevin, V. Kalogera, and A. K. Katsaggelos, ArXiv e-prints (2017), arXiv:1705.00034 [cs.LG].

[76] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[77] K. Simonyan and A. Zisserman, CoRR **abs/1409.1556** (2014).

[78] K. He, X. Zhang, S. Ren, and J. Sun, CoRR **abs/1512.03385** (2015).

[79] B. J. Owen and B. S. Sathyaprakash, *Phys. Rev.* D **60**, 022002 (1999).

[80] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks: 17th International Conference, EANN 2016, Aberdeen, UK, September 2-5, 2016, Proceedings*, edited by C. Jayne and L. Iliadis (Springer International Publishing, Cham, 2016) pp. 213–226.

[81] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Web-Age Information Management: 15th International Conference, WAIM 2014, Macau, China, June 16-18, 2014. Proceedings*, edited by F. Li, G. Li, S.-w. Hwang, B. Yao, and Z. Zhang (Springer International Publishing, Cham, 2014) pp. 298–310.

[82] M. Pürrer, *Phys. Rev.* D **93**, 064041 (2016), arXiv:1512.02248 [gr-qc].

[83] K. Belczynski, S. Repetto, D. Holz, R. O'Shaughnessy, T. Bulik, E. Berti, C. Fryer, and M. Dominik, ArXiv e-prints (2015), arXiv:1510.04615 [astro-ph.HE].

[84] F. Löffler, J. Faber, E. Bentivegna, T. Bode, P. Diener, R. Haas, I. Hinder, B. C. Mundim, C. D. Ott, E. Schnetter, G. Allen, M. Campanelli, and P. Laguna, Classical and Quantum Gravity **29**, 115001 (2012), arXiv:1111.3344 [gr-qc].

[85] D. Shoemaker, "Advanced LIGO anticipated sensitivity curves," (2010).

[86] A. H. Mroué, M. A. Scheel, B. Szilágyi, H. P. Pfeiffer, M. Boyle, D. A. Hemberger, L. E. Kidder, G. Lovelace, S. Ossokine, N. W. Taylor, A. Zenginoğlu, L. T. Buchman, T. Chu, E. Foley, M. Giesler, R. Owen, and S. A. Teukolsky, Physical Review Letters **111**, 241104 (2013), arXiv:1304.6077 [gr-qc].

[87] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, edited by G. B. Orr and K.-R. Müller (Springer Berlin Heidelberg, 1998) pp. 9–50.

[88] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Proceedings of the IEEE **86**, 2278 (1998).

[89] S. Ioffe and C. Szegedy, CoRR **abs/1502.03167** (2015).

[90] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Journal of Machine Learning Research **15**, 1929 (2014).

[91] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, CoRR **abs/1512.01274** (2015).

[92] L. Perreault Levasseur, Y. D. Hezaveh, and R. H. Wechsler, ArXiv e-prints (2017), arXiv:1708.08843.

[93] T. Chu, H. Fong, P. Kumar, H. P. Pfeiffer, M. Boyle, D. A. Hemberger, L. E. Kidder, M. A. Scheel, and B. Szilagyi, Classical and Quantum Gravity **33**, 165001 (2016), arXiv:1512.06800 [gr-qc].

[94] E. A. Huerta, P. Kumar, B. Agarwal, D. George, H.-Y. Schive, H. P. Pfeiffer, R. Haas, W. Ren, T. Chu, M. Boyle, D. A. Hemberger, L. E. Kidder, M. A. Scheel, and B. Szilagyi, *Phys. Rev.* D **95**, 024038 (2017).

[95] E. A. Huerta, P. Kumar, S. T. McWilliams, R. O'Shaughnessy, and N. Yunes, *Phys. Rev.* D **90**, 084016 (2014), arXiv:1408.3406 [gr-qc].

[96] E. A. Huerta and D. A. Brown, *Phys. Rev.* D **87**, 127501 (2013), arXiv:1301.1895 [gr-qc].

[97] "The Wolfram Language Image Identification Project," https://www.imageidentify.com/.

[98] S. Ioffe and C. Szegedy, in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)* (2015) pp. 448–456.

[99] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, arXiv preprint arXiv:1706.02677 (2017).

[100] Y. Bengio and Y. LeCun, in *Large Scale Kernel Machines*, edited by L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (MIT Press, 2007).

[101] V. Tiwari, S. Klimenko, N. Christensen, E. A. Huerta, S. R. P. Mohapatra, A. Gopakumar, M. Haney, P. Ajith, S. T. McWilliams, G. Vedovato, M. Drago, F. Salemi, G. A. Prodi, C. Lazzaro, S. Tiwari, G. Mitselmakher, and F. Da Silva, *Phys. Rev.* D **93**, 043007 (2016), arXiv:1511.09240 [gr-qc].

[102] T. Zeng and S. Ji, in *2015 IEEE International Conference on Data Mining* (2015) pp. 579–588.

[103] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, CoRR **abs/1610.00087** (2016).

[104] Y. Xu, J. Du, L. R. Dai, and C. H. Lee, IEEE/ACM Transactions on Audio, Speech, and Language Processing **23**, 7 (2015).

[105] A. Kumar and D. Florêncio, CoRR **abs/1605.02427** (2016).

[106] H. J. Pletsch and B. Allen, Physical Review Letters **103**, 181102 (2009), arXiv:0906.0023 [gr-qc].

[107] "GPU-Based Deep Learning Inference: A Performance and Power Analysis," https://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf.

[108] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, SIGARCH Comput. Archit. News **44**, 243 (2016).

[109] G. E. Hinton and R. R. Salakhutdinov, Science **313**, 504 (2006).

[110] T. Piran, E. Nakar, and S. Rosswog, *MNRAS* **430**, 2121 (2013), arXiv:1204.6242 [astro-ph.HE].

[111] W. H. Lee, E. Ramirez-Ruiz, and G. van de Ven, *Astrophys. J.* **720**, 953 (2010).

[112] W. H. Lee and E. Ramirez-Ruiz, New Journal of Physics **9**, 17 (2007), astro-ph/0701874.

[113] N. Sedaghat and A. Mahabal, ArXiv e-prints (2017), arXiv:1710.01422 [astro-ph.IM].

[114] P. Amaro-Seoane, S. Aoudia, S. Babak, P. Binétruy, E. Berti, A. Bohé, C. Caprini, M. Colpi, N. J. Cornish, K. Danzmann, J.-F. Dufaux, J. Gair, O. Jennrich, P. Jetzer, A. Klein, R. N. Lang, A. Lobo, T. Littenberg, S. T. McWilliams, G. Nelemans, A. Petiteau, E. K. Porter, B. F. Schutz, A. Sesana, R. Stebbins, T. Sumner, M. Vallisneri, S. Vitale, M. Volonteri, and H. Ward, Classical and Quantum Gravity **29**, 124016 (2012), arXiv:1202.0839 [gr-qc].

[115] J. R. Gair, M. Vallisneri, S. L. Larson, and J. G. Baker, Living Reviews in Relativity **16**, 7 (2013), arXiv:1212.5575 [gr-qc].

[116] A. Sesana, Physical Review Letters **116**, 231102 (2016), arXiv:1602.06951 [gr-qc].

[117] R. Pascanu, T. Mikolov, and Y. Bengio, in *ICML (3)*.

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

[119] A. Graves, A. Mohamed, and G. E. Hinton, CoRR **abs/1303.5778** (2013).

[120] L. K. Nuttall, T. J. Massinger, J. Areeda, J. Betzwieser, S. Dwyer, A. Effler, R. P. Fisher, P. Fritschel, J. S. Kissel, A. P. Lundgren, D. M. Macleod, D. Martynov, J. McIver, A. Mullavey, D. Sigg, J. R. Smith, G. Vajente, A. R. Williamson, and C. C. Wipf, Classical and Quantum Gravity **32**, 245005 (2015), arXiv:1508.07316 [gr-qc].