Efficient gravitational-wave glitch identification from environmental data through machine learning

Robert E. Colgan,[1,2] K. Rainer Corley,[3,4] Yenson Lau,[2,5] Imre Bartos,[6] John N. Wright,[2,5] Zsuzsa Márka,[4] and Szabolcs Márka[3]

[1]*Department of Computer Science, Columbia University in the City of New York, 500 West 120th Street, New York, New York 10027, USA*
[2]*Data Science Institute, Columbia University in the City of New York, 550 West 120th Street, New York, New York 10027, USA*
[3]*Department of Physics, Columbia University in the City of New York, 538 West 120th Street, New York, New York 10027, USA*
[4]*Columbia Astrophysics Laboratory, Columbia University in the City of New York, 538 West 120th Street, New York, New York 10027, USA*
[5]*Department of Electrical Engineering, Columbia University in the City of New York, 500 West 120th Street, New York, New York 10027, USA*
[6]*Department of Physics, University of Florida, P.O. Box 118440, Gainesville, Florida 32611-8440, USA*

(Received 5 February 2020; accepted 1 May 2020; published 20 May 2020)

The LIGO observatories detect gravitational waves through monitoring changes in the detectors' length down to below 10^{-19} m/$\sqrt{\text{Hz}}$ variations—a small fraction of the size of the atoms that make up the detector. To achieve this sensitivity, the detector and its environment need to be closely monitored. Beyond the gravitational-wave data stream, LIGO continuously records hundreds of thousands of channels of environmental and instrumental data in order to monitor for possibly minuscule variations that contribute to the detector noise. A particularly challenging issue is the appearance in the gravitational wave signal of brief, loud noise artifacts called "glitches," which are environmental or instrumental in origin but can mimic true gravitational waves and therefore hinder sensitivity. Currently, they are primarily identified by analysis of the gravitational-wave data stream, and auxiliary data channels often provide corroborating evidence. Here we present a machine learning approach that can identify glitches by considering *all* environmental and detector data channels, a task that has not previously been pursued due to its scale and the number of degrees of freedom within gravitational-wave detectors. The presented method is capable of reducing the gravitational-wave detector network's false alarm rate and improving the LIGO instruments, consequently enhancing detection confidence.

DOI: 10.1103/PhysRevD.101.102003

I. INTRODUCTION

Modern interferometric gravitational-wave (GW) detectors [1,2] are highly complex and sensitive instruments. Each detector is sensitive not only to gravitational radiation, but also to noise from sources including the physical environment, seismic activity, and complications in the detector itself. The output data of these detectors are therefore also highly complex. In addition to the desired signal, the GW data stream contains sharp lines in its noise spectrum and non-Gaussian transients, or "glitches," that are not astrophysical in origin.

Instrumental artifacts in the GW data stream can be mistaken for short-duration, unmodeled GW events, and such non-Gaussian noise can also decrease the confidence in compact binary detections, sometimes by orders of magnitude [3]. We show an example of the similarity between a glitch and a GW signal in Fig. 1 to illustrate the difficulty in searching for GW signals with glitches present. Thus, it is important to identify and flag GW data containing glitches. Flagged instrumental glitches can then be addressed in many ways, from graceful modeling followed by subtraction to the cruder approach of so-called "vetoes" that unnecessarily waste data. Understanding the origin of instrumental glitches is also important for diagnosing their causes and improving the quality of the detector and its data.

The primary pipeline currently used to identify and characterize glitches in the Advanced LIGO and Virgo detectors is Omicron [7] (see also Refs. [4,5]). Omicron identifies glitches by searching the GW strain data from a single detector for events of excess power. It characterizes their properties, such as amplitude, duration, and frequency, by comparing the event to a sine-Gaussian waveform.

In addition to the GW data stream, each detector records hundreds of thousands of "channels" of auxiliary data, each channel measuring some aspect of the detector's components

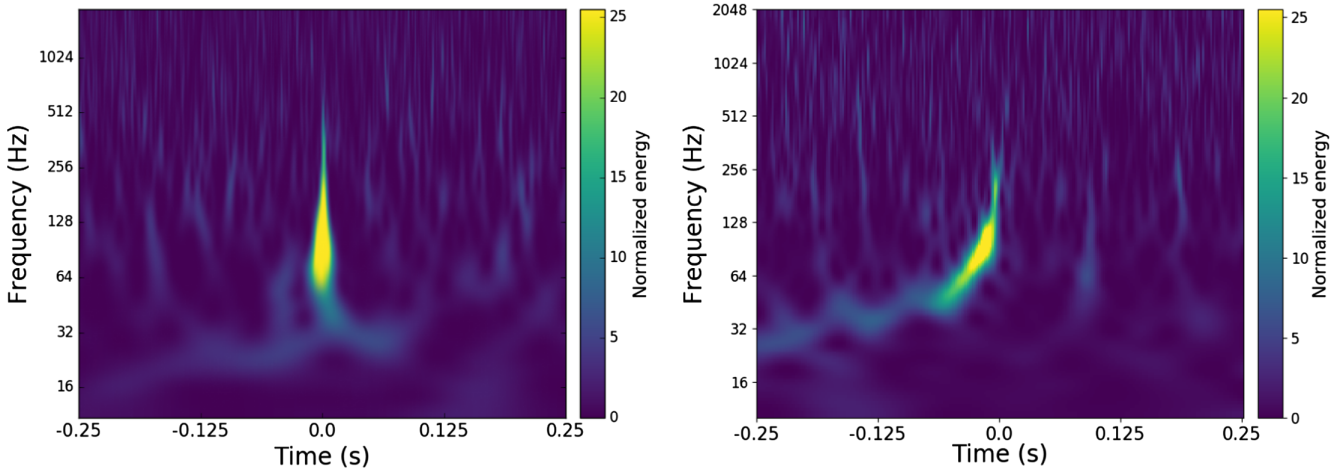

FIG. 1. Comparison of the omegagrams [4,5]—a Gabor-wavelet time-frequency representation of the strain data—of a glitch event (left) and a GW event (right) in LIGO O2 data. Glitches can be mistaken for unmodeled GW events and also trigger modeled searches. Images obtained from GravitySpy [6] database.

or physical environment [8]. These channels provide important information about the state of the detector that can be useful for diagnosing glitches, but monitoring all of them is a difficult task.

When identifying and flagging potential glitches, it is important to ensure that the event is indeed a glitch of instrumental origin, rather than an unmodeled GW event. By considering only LIGO's auxiliary channels, rather than the GW data stream, we can be more confident that glitches we flag are indeed not from gravitational events, since the auxiliary channels are generally not sensitive to GWs [9,10]. An additional benefit of studying glitches with auxiliary channels is that correlations with specific channels can help identify the source of detector issues.

Current work to veto glitch segments includes UsedPercentageVeto [10], HierarchicalVeto [11], Bilinear Coupling Veto [12], iDQ [13–15], and GravitySpy's [6] classification information can also be used in offline veto studies. Cavaglià *et al.* [16] use a dual approach of random forests combined with genetic programming in order to identify instrumental artifacts from a subset of auxiliary channels. Additionally, there is work to study data quality by correlating auxiliary channels with the GW detector's astronomical range [17]. Our method is complementary to these approaches; we consider *all* auxiliary channels and use them to identify segments containing glitches, both to flag these segments and to identify detector issues.

Machine learning techniques have proved to be powerful tools in analyzing complex problems by learning from large example datasets. They have been applied in GW science from as early as 2006 [18] to the study of glitches [6,13,14,16,19–23] and other problems, such as real-time signal detection [24], signal characterization [25–31], and parameter estimation [32–34].

Classification is a fundamental problem in machine learning in which a machine learning model is trained to consider a set of measured characteristics, or "features," of data samples belonging to one of at least two categories. By providing the model with a set of samples whose categories are known, we can train it to predict the category of samples whose true category is unknown.

We pose the problem of detecting whether a glitch is occurring at a given time based on LIGO auxiliary channels as a simple two-class classification problem (as in previous work [14,16]) and apply a well-understood, efficient, and commonly used machine learning method to this problem, with promising results.

We train a classification model to predict whether a glitch is occurring at a given time using features derived from the GW detector's auxiliary channels at that time. Because the data on which the classification is performed are derived only from the auxiliary channels, our method is able to provide corroboration of the presence or absence of glitches without using strain data, independently of existing methods that analyze the strain.

Below, we describe a variant of the method, called Elastic-net-based Machine learning for Understanding (EMU method or method hereafter). We describe the method in detail and how we use it to identify glitches in Sec. II; we show the results of testing this method on recent LIGO data in Sec. III; and we discuss the results in Sec. IV. We developed and tested the method with recent data from LIGO Livingston Observatory (LLO) and glitch information reported by Omicron, but the method is applicable to any current and future observatories that record comprehensive auxiliary data. Furthermore, the method might also be applicable to diagnostics of complex systems outside of gravitational-wave science. Note that because the performance measure examples presented here rely on Omicron during training to learn characteristics associated with glitches, they contain an implicit dependence on Omicron's extremely broad determination of what

constitutes a glitch. However, the use of Omicron can easily be replaced or complemented with any other event trigger generator that reports glitch time information.

II. METHODS

To predict the presence of a glitch in a GW data stream using auxiliary information, we need a method that extracts useful information from the auxiliary channels and uses this information to make a decision.

In considering all of the auxiliary channels, we must generate and process a large, high-dimensional dataset for classification. We choose as the feature set a group of representative statistics for each auxiliary channel to capture properties of the channel's behavior in the vicinity of a glitch (or absence of one). Linear models are simple and effective on this type of large dataset, and they are straightforward to train.

We use logistic regression, a standard model for binary classification. It is a member of the family of generalized linear models, which are efficient to train and test: the number of trained parameters is equal to the dimension of the input, unlike models such as neural networks, and the optimization problem is convex, lending itself to efficient iterative methods [35,36]. Additionally, the sparsity of the trained model caused by the use of elastic net regularization [37] results in an even more efficient model during testing, where the number of multiplications required is only the number of nonzero coefficients in the model (less than 100 in our case—see Sec. II E).

If we hope to also diagnose detector issues, it is important that we can interpret the output of the algorithm. Logistic regression provides a simple method for this: during training, each feature is given a weight; higher-magnitude weights indicate features that are more relevant in deciding whether a glitch is present in a sample. A model with fewer large weights is intuitively more easily interpretable. To encourage this property, we use elastic net regularization to penalize the weights such that only the most relevant features are selected by the model, and those corresponding to uninformative features become 0. This also allows us to train with smaller datasets than would otherwise be required if we did not impose sparse regularization.

Below, we describe the preliminary step of identifying irrelevant data in Sec. II A, the extraction of features from the remaining data in Sec. II B, preconditioning of the data in Sec. II C, the machine learning model we employ in Sec. II D, the selection of model hyperparameters in Sec. II E, and an analysis of how much training data is sufficient in Sec. II F.

A. Preliminary data reduction

There are approximately 250 000 auxiliary channels in each LIGO detector, with sample rates of 16 Hz and higher. Many of these channels are constant or always change in a consistent pattern (e.g., tracking the time or counting CPU cycles) and contribute no actionable information to the classification model. To improve the efficiency of the training process, the first step of our method is to identify such uninformative channels so we may ignore them in the subsequent training step. (If they had been included, the training process would learn to set all of their associated coefficients to zero because they contain no actionable information, so they would subsequently be ignored anyway, and it would cost us carbon credits due to wasted CPU cycles.)

Auxiliary channel time-series data is encoded in a custom format [38] and stored in files each containing 64 consecutive seconds of data for each channel; we refer to these files as "raw frame files." To identify uninformative channels, we choose a few raw frame files from the training period of each analysis and compare the channels across those frames. For each channel in each of the selected frames, we subtract the channel's first raw value from the following values in that frame, and compare the resulting time series to the corresponding one from each of the other selected frames. If all are identical, the channel is ignored for the rest of the analysis. After this procedure, approximately 40 000 channels remain for our further analysis.

Some auxiliary channels are directly coupled with the GW strain channel, or are contaminated in other ways by the GW signal. Many of these channels have been identified in internal LIGO "channel safety" studies [9,10]. We removed all such known channels. Additionally, after a preliminary training run, we consulted with site experts to identify whether any of the channels selected by the model might be contaminated by the strain but not yet considered in safety studies. In an effort to be conservative and demonstrate that our performance was not influenced by channels that may be coupled to the strain, we removed all such channels from further analyses.

B. Feature extraction

We use the Omicron [7] event trigger generator to identify glitch times for training and testing our model. (Once trained, our model is fully independent of Omicron and the strain data; it considers only parameters computed from auxiliary channels, as described below.) Omicron analyzes low-latency strain data to find events of excess power and reports parameters of these glitches, including start time, peak time, and duration.

For our training, we gather points in time (i) drawn from the peak times of glitches ("glitchy" times), and (ii) drawn from stretches of at least four seconds with no recorded glitches ("glitch-free" times). For the glitch-free samples, we select times such that no part of any glitch (accounting for its full duration) falls within 2 seconds before or after the sample time. (We note that this does not mean we veto 4 seconds of data for each potential glitch our method flags;

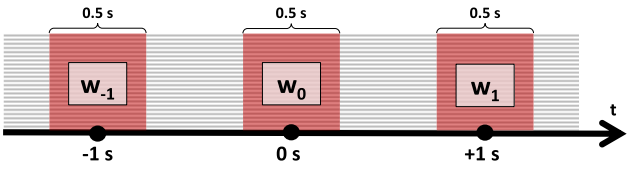

FIG. 2. Illustration of the time intervals considered in statistical feature array.

it simply means that during the training process, we require the time segment around our glitch-free training examples to be sufficiently clean.)

For each glitchy or glitch-free point in time, we generate an array of ten statistical quantities (described in the following paragraphs) for each channel to characterize the channel's behavior around that time. These quantities become the features for our analysis.

Let us denote a given glitch peak time or glitch-free sample time as t_0 and the times 1 second before and 1 second after as t_{-1} and t_1, respectively. We consider three time windows of 0.5 seconds duration centered at t_{-1}, t_0, and t_1, denoted w_{-1}, w_0, and w_1. This is illustrated in Fig. 2. Note that the lowest-frequency channels in LLO at the time of our experiments have a sample rate of 16 Hz, so each window contains a minimum of eight samples.

For a given sample time t_0, for each channel we construct the following ten-dimensional vector based on the three time windows:

$$\mathbf{v} = (\mu_{-1}, \quad \mu_0, \quad \mu_1, \quad \sigma_{-1}, \quad \sigma_0, \quad \sigma_1, \tag{1a}$$

$$\mu_1 - \mu_{-1}, \quad \sigma_1 - \sigma_{-1}, \tag{1b}$$

$$\mu_0 - \frac{\mu_1 + \mu_{-1}}{2}, \quad \sigma_0 - \frac{\sigma_1 + \sigma_{-1}}{2}), \tag{1c}$$

where μ_i and σ_i are the mean and standard deviation, respectively, of the channel's raw value over the time window w_i, with $i \in \{-1, 0, 1\}$. (We note that this does not mean we veto 2.5 seconds of data for each potential glitch our method flags; it simply means that for a given channel and sample time, we consider the surrounding time period so we can identify deviations from the channel's behavior in its local environment, as implied in the definition of the features.) The time and duration of the transient can be extracted as the feature definition is known.

In contrast to Ref. [14], which considers auxiliary data in a 100 millisecond window around glitch transients, our method considers data from a span of 2.5 seconds, enabling longer-timescale couplings to be addressed. We also consider many more channels in our analysis.

Each feature was chosen with the intent of capturing certain properties of a channel's behavior in the vicinity of a glitch or the absence of one. The features in Eq. (1a) were chosen to capture the mean and standard deviation of the channel's raw value shortly before, during, and shortly after t_0. Those in Eq. (1b) were chosen to identify step changes occurring near t_0. Those in Eq. (1c) were chosen to identify short, temporary changes occurring near t_0.

It should be noted that these features and the lengths of the time windows were chosen *ad hoc* based on intuition of what properties of channels' behavior might be informative. We acknowledge that this choice may limit the ability of this method to uncover glitches which have significantly different timescales. However, compared to extracting features from nearby auxiliary transients in various wavelet domains (see, e.g., Refs. [14,16]), the method presented here is simpler and requires fewer computational resources, especially when considering the significantly increased dimensionality of the problem addressed here. We believe the simplicity of these features is an advantage of our method, but it is otherwise essentially agnostic to the features chosen here; more descriptive features could potentially improve its performance. We leave this exploration to future work.

We construct the vector $\mathbf{v}$ for each of the approximately 40 000 channels in consideration, resulting in approximately 400 000 features for each glitchy or glitch-free point in time.

C. Data preconditioning

Most machine learning techniques assume that each feature is on approximately the same scale; otherwise, features whose raw values are large in magnitude would dominate the others. A standard normalization procedure is to replace raw values with their standard score (i.e., the number of standard deviations away from the training mean that the raw value falls), so each feature has zero mean and unit standard deviation over the training set [35,36]. For each analysis under consideration, we compute the mean and standard deviation over the training set; then, for every point in the training, validation, and test sets, we subtract the mean and divide by the standard deviation of that feature in the training set.

Occasionally, the raw channel data contain missing or invalid values, resulting in invalid entries in our feature matrix. When this occurs, as is standard practice [39], we simply replace the entry with the mean of the valid entries for that feature in the training set prior to performing the normalization described above. Because the data are normalized to zero mean, this effectively results in that entry subsequently being ignored by the model. (The alternative of discarding the entire entry or channel would unnecessarily waste potentially valuable data.)

D. Glitch classification via logistic regression

We formulate the problem of identifying glitches as a basic statistical classification problem, where instances in which a glitch is present are classified as 1 ("glitchy"), and instances where no glitch is present are classified as 0 ("glitch-free").

We use logistic regression with elastic net regularization to perform this classification using the features derived from auxiliary channel data described in Sec. II B.

Logistic regression is a well-established linear classification method in statistics and machine learning [35,36]. It is related to classical linear regression, but rather than predicting a continuous unbounded variable, a logistic function is applied to the output to restrict it between 0 and 1.

Given a set of n training data points in p dimensions (i.e., each data point has p features) and n corresponding binary labels (the ground truth), a logistic regression model is trained by iteratively minimizing the residual error between the predicted class probability of the training data and ground truth. The trained model consists of a set of p coefficients (or "weights") $\mathbf{w}$ and a bias term b; the dot product of these coefficients and a test data point plus the bias term is passed through a logistic function to obtain an estimate of the probability that the point should be classified 0 or 1.

Let $\sigma(\cdot)$ denote the logistic function

$$\sigma(a) = \frac{1}{1+\exp(-a)}. \tag{2}$$

Then, the probability estimated by the model that a test data point $\mathbf{x}$ belongs to the class 1 is

$$P(\mathbf{x}=1) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b). \tag{3}$$

This value may be thresholded to produce a binary output. The threshold is commonly 0.5 but may be chosen as desired to adjust the ratio of false positives and false negatives.

During training, a measure of the error between the known ground truth label $y_i \in \{0,1\}$ and the current model's prediction $\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)$ for a training point $\mathbf{x}_i \in \mathbb{R}^p$ can be quantified:

$$\begin{aligned} E_{\mathbf{w},\mathbf{b}}(y_i,\mathbf{x}_i) = -(&y_i \log(\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)) \\ &+ (1-y_i)\log(1-\sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b))). \end{aligned} \tag{4}$$

Known as the cross-entropy error, this is a convex function that can be minimized over $\mathbf{w}$ and b by gradient descent or other iterative methods [35,36].

Various regularization terms may be applied to the coefficients and added to the residual error during training as a penalty, to reduce overfitting and induce desired properties in the trained model [35,36]. Let $R(\mathbf{w})$ be some regularization function for the coefficients $\mathbf{w}$. The combined cost (or "loss") function that is iteratively minimized during training is given by

$$L(\mathbf{w},b) = \frac{1}{n}\sum_{i=1}^{n} E_{\mathbf{w},b}(y_i,\mathbf{x}_i) + \alpha R(\mathbf{w}), \tag{5}$$

where α is a hyperparameter controlling the overall regularization strength relative to the error term E.

Common choices of regularization functions with logistic regression are the L2 norm (also known as ridge regression or Tikhonov regularization) and the L1 norm (the same penalty used in the LASSO [40]). In addition to mitigating overfitting by penalizing the overall magnitude of the coefficient vector, the L1 norm also induces sparsity in the coefficients (i.e., many of them will be zero). This is often desirable for scalability and interpretability when the dimension p of the input data is high, as is the case with our dataset. After training, the nonzero coefficients suggest which of the input features are most important in determining the classification result [36].

The elastic net [37] is a weighted sum of the L1 and L2 norms:

$$R(\mathbf{w}) = \frac{\lambda}{2}\sum_{j=1}^{p} w_j^2 + (1-\lambda)\sum_{j=1}^{p}|w_j|, \tag{6}$$

where λ is a hyperparameter controlling the relative strengths of the L1 and L2 regularizations. The elastic net also induces sparsity, but less strongly than L1. It also has the desirable advantage over L1 of being more likely to select correlated features together rather than arbitrarily choosing only one of them [37]. In our application, if features from many channels are correlated, it may be useful for diagnostic purposes to consider all of them rather than only one.

E. Hyperparameter optimization

Elastic net logistic regression has two hyperparameters that must be tuned to achieve the best result: overall regularization strength α, in Eq. (5), and the ratio λ of the strengths of L1 and L2 regularization, in Eq. (6).

We performed a grid search across a range of both parameters using data from LLO during ER14 and evaluated the results on a held-out validation dataset drawn from a separate period of time (see Fig. 3). Using a validation dataset separate from both the training set and the test set on which we report our results allows us to choose hyperparameters that will generalize well to unseen data without overfitting to our training or test data [35,36].

We trained models over a grid of α and λ values on a dataset drawn from 30 000 seconds during a lock segment on 6 March 2019 (GPS time 1 235 870 000 to 1 235 900 000). To create the training dataset, we randomly sampled 7500 glitch-free points in time (as described in Sec. II B) and 7500 of the 30 141 Omicron glitches during that period. This subsampling was performed to allow the dataset to fit in available memory. For both datasets, we

FIG. 3. Time periods used to create training, validation, and test datasets. The ER14 analysis includes a validation dataset for tuning hyperparameters, which were fixed for all subsequent tests, so there is no validation period for the O3 analysis.

also ignored any samples falling too close to the beginning or end of a 64-second raw frame file. (This represents less than 4% of the data, and was done for technical convenience; it is not inherent to the method.) After preliminary data reduction as described in Sec. II A, the number of channels considered was 38 235. We then generated the 382 350 statistical features for each point, as described in Sec. II B. We then trained an elastic net logistic regression model independently for each α, λ pair on this training set. Training was performed using the Scikit-learn package [39].

We evaluated each trained model on a validation dataset drawn from the 10 000 seconds immediately following the training period (GPS time 1 235 910 000 to 1 235 920 000). Figure 3 illustrates the times these ER14 datasets are drawn from. The validation dataset was created similarly to the training set, by sampling 2500 glitch-free points and 2500 of the 7222 Omicron glitches during that period. We chose the α, λ pair that gave the best accuracy on the validation dataset and fixed the values of those hyperparameters based on these results for all further training. (Although a small fraction of the channels were removed and others added between ER14 and O3, the hyperparameters do not depend specifically on the channels and generalize well.)

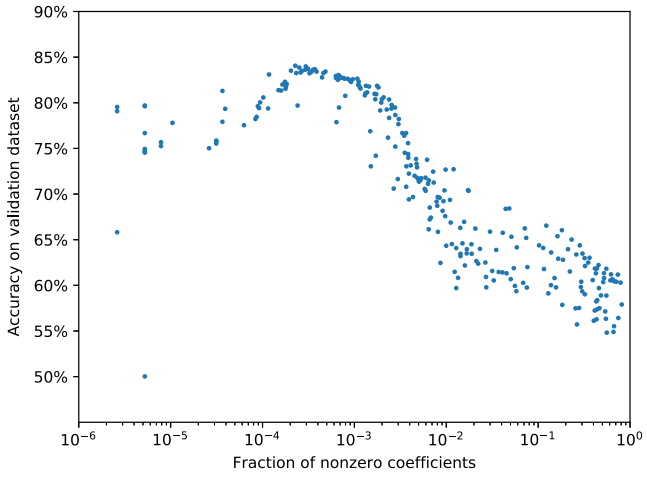

FIG. 4. ER14 validation accuracy vs fraction of nonzero coefficients in the trained model. The models have approximately $p = 400\,000$ coefficients, so a model with $10^{-3}p$ nonzeros contains only 400 nonzero coefficients.

Varying the α and λ hyperparameters effectively tunes the strength with which the model's coefficients w_j are driven to zero by regularization during training. After training with many pairs of these parameters, we can evaluate the relationship between the number of nonzero coefficients and the model's predictive accuracy on the validation data. One would expect that a model with too many zero coefficients would not be able to consider enough features to make accurate predictions, while a model with too many nonzero coefficients would become overfit to the training data and not generalize well to separate data. Figure 4 demonstrates that this is the case with our data.

The model that achieves the highest accuracy on our validation dataset, at 84.1%, contains only 87 (0.02%) nonzero coefficients. These coefficients correspond to features from only 56 distinct channels, indicating specific channels of potential detector issues at training time. None of these 56 channels are known to be coupled with or contaminated by the GW strain. This demonstrates that a small subset of the features and channels is sufficient to consider for effective glitch identification performance. It also demonstrates that elastic net regularization is beneficial not only to interpretability but also to prediction accuracy, and that the method is robust to uninformative, unactionable, and faulty channels. Note that we do not claim that these are the *only* channels associated with glitches—merely that these are the ones for which the association is strong enough in this dataset to be discovered by this model, and that they are sufficient to achieve a useful level of performance. Training on different segments of data will naturally produce different numbers of significant channels, potentially with some overlap, as we observed with the ER14 and O3 results described in Sec. III.

F. Amount of training data

We also investigated how much training data was sufficient for good performance, using the ER14 training and validation data. For this experiment, we trained a new classifier using data drawn from time periods of varying

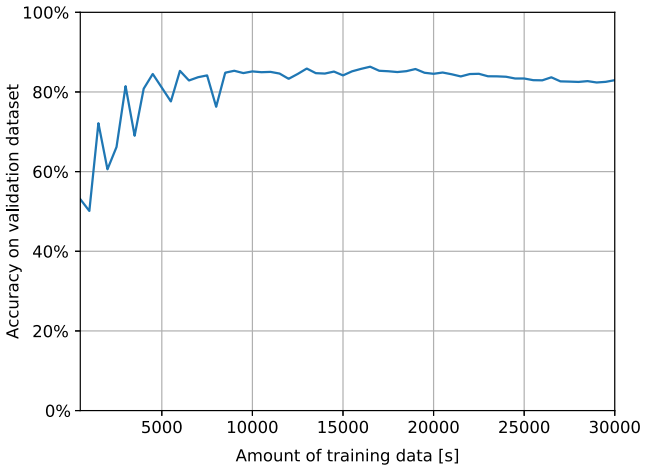

FIG. 5. Classifier's accuracy vs length of training period.

lengths. Each of these time periods was a subset of the original training data period, ending at the end of the original training period and starting between 500 and 30 000 seconds earlier. We show the accuracy of the classifier on the validation dataset (described in Sec. II E) for these varying training lengths in Fig. 5. The results indicate that 10 000 seconds is a sufficient length of time from which to draw training data. They also suggest that using too long of an interval of training data may actually hurt performance, likely because the detector's state drifts over time and the model is not flexible enough to account for this behavior while maintaining fine-grained accuracy for more specific instances in time.

III. RESULTS

We use the EMU method described above to classify glitchy and glitch-free times for several recent segments of data from LLO. We show two test cases from different, recent runs: one from a lock segment during ER14, and a second from a lock segment during O3.

The classifier model for each analysis was trained independently using data drawn from a time period close to but not overlapping with the time period from which the test dataset for that analysis was drawn. We do not perform hyperparameter optimization as we did for ER14 in Sec. II E again for O3, because the procedure is computationally intensive, and the similar nature of the data means the optimal hyperparameters would likely not be significantly different.

For each analysis, we create a test dataset drawn from a period of time separate from the training dataset (and, in the case of ER14, separate from the validation dataset as well), as illustrated in Fig. 3. We standardize the test dataset according to the means and standard deviations of the features in the training dataset. We then pass the test dataset through the classifier without labels and compare the predictions to the known ground truth for each point. For each incorrect classification, we can specify whether the result is a false positive (a glitch-free time classified as glitchy) or a false negative (a glitchy time classified as glitch-free).

Note that the actual output of the classifier is the predicted probability of a glitch, which ranges between 0 and 1. Prior to calculating all reported accuracies, we threshold this value at 0.5, so values at or above are considered predictions of glitches and values below are considered predictions of the absence of a glitch. We can adjust this threshold to control the ratio of true positives and false positives as necessary for different applications, which might call for a lower false negative rate at the expense of a higher false positive rate or vice versa. The trade-off is illustrated for both analyses in the Receiver Operating Characteristic (ROC) curve in Fig. 6.

At the default decision threshold of 0.5, the accuracy on the ER14 test dataset is 83.8%, with a true positive rate of 73.0% and a true negative rate of 94.6%. The accuracy on the O3 test dataset is 79.9%, with a true positive rate of 62.1% and a true negative rate of 97.7%. We also show the overall accuracy, true positive rate, and true negative rate over time during the test periods for each analysis in Figs. 7 and 8.

If we restrict our analysis in training and testing to glitches with a signal-to-noise ratio (SNR) at or above 6, we achieve an overall accuracy of 88.2% for ER14 and 90.3% for O3, with true positive rates of 80.8% and 86.7% and true negative rates of 95.4% and 93.8%, respectively. (The minimum SNR reported by Omicron is 5; roughly 30% of glitches have an SNR at or above 6.) The corresponding ROC curves are displayed in Fig. 6.

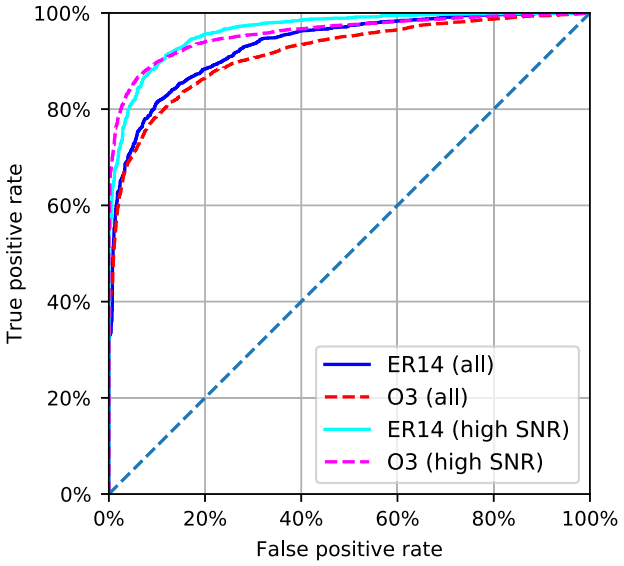

FIG. 6. Overall ROC curves for ER14 and O3 test segments, and corresponding ROC curves for only high-SNR glitches.

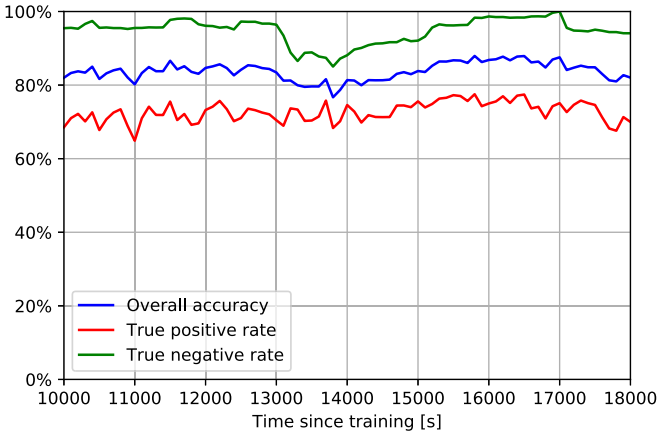

FIG. 7. Accuracy, true positive rate, and true negative rate over time during the ER14 test period. For stability, these quantities are computed over rolling windows of length 2000 s beginning at the indicated time. The x axis begins at 10 000 s, because the 0 to 10 000 s period is used for validation (see Fig. 3).

A. ER14

For the ER14 test analysis, we use the trained classifier model that performed best on the ER14 validation dataset used for hyperparameter optimization, as described in Sec. II E. The training dataset is therefore the same as described there. Recall that to create this dataset, we sampled 7500 glitch-free points in time (as described in Sec. II B) and 7500 of the 30 141 Omicron glitches during the period between GPS times 1 235 870 000 and 1 235 900 000. After preliminary data reduction as described in Sec. II A, the number of channels considered was 38 235, so the training and test datasets have 382 350 features.

The nonzero coefficients of a trained classifier indicate which features the classifier considers when making decisions. As discussed in Sec. II E, the classifier that performed best on the ER14 validation data had 87 nonzero coefficients corresponding to features from 56 different channels.

The test data are drawn from 9616 seconds between GPS times 1 235 910 000 and 1 235 919 616. We sampled 2500 glitch-free points in time and 2500 of the 6479 Omicron glitches during that period, chosen at random. As with the training, we also ignored any samples falling too close to the beginning or end of a 64-second raw frame file. The classifier achieves an accuracy of 83.8% on this test dataset, with a true positive rate of 73.0% and a true negative rate of 94.6%.

The classifier's accuracy, true positive rate, and true negative rate over time in the test period is shown in Fig. 7. This result indicates that the performance is relatively consistent over time, but may be affected by transient changes in the state of the detector that were not seen during training.

B. O3

For the O3 analysis, we trained a new classifier on a training dataset drawn from 10 000 seconds during 10 April 2019 (GPS times 1 238 900 000 to 1 238 910 000). We used a smaller amount of time for the training period for this analysis because the results shown in Fig. 5 indicate that 10 000 seconds of training data is sufficient for good performance. Similarly to the ER14 training dataset, we sampled 2500 glitch-free points in time and 2500 of the 8098 Omicron glitches during that period, chosen at random, and we ignored any samples falling too close to the beginning or end of a 64-second raw frame file. After preliminary data reduction as described in Sec. II A, the number of channels considered is 38 327, so the training and test datasets have 383 270 features.

After training, the O3 classifier had 55 nonzero coefficients corresponding to features from 46 distinct channels.

The test data are drawn from 30 000 seconds between GPS times 1 238 910 000 and 1 238 940 000. We sampled 7500 glitch-free points in time and 7500 of the 24 243 Omicron glitches during that period, chosen at random, and we ignored any samples falling too close to the beginning or end of a 64-second raw frame file. The classifier achieves an accuracy of 79.9% on this test dataset, with a true positive rate of 62.1% and a true negative rate of 97.7%.

The accuracy, true positive rate, and true negative rate over time during the test period are shown in Fig. 8. As with ER14, this result indicates that the performance is relatively consistent over time, but there is a visible dip in true negative rate and a corresponding dip in accuracy soon after the beginning of the test period, suggesting some nonstationarity in the data or the state of the detector. This is not surprising, especially since the model was trained on only about 3 hours of data; it should be taken into account that this method would likely become increasingly susceptible to such issues the less training data it sees and the more time has passed since training. One simple way to mitigate this would be to periodically retrain the model on recent data.

To illustrate one way this method might be used in practice, we also performed an experiment in which we recorded our model's estimate of glitch probability over a continuous segment of time and compared the output to the actual locations of glitches during that segment. For this experiment, we used 64 seconds of data during O3 beginning at GPS time 1 238 900 480. The results of this experiment are shown in Fig. 9, illustrating that the classifier's output largely does correctly indicate the presence or absence of a glitch over time. If the trained classifier were fed the auxiliary channel data continuously in this fashion, it could potentially be used in near real time to corroborate the likelihood that a potential event appearing in the strain at any given time is astrophysical or instrumental in origin, independently of other existing two-class glitch classification systems used as an automatic, low-latency response to candidate events [15]. By considering

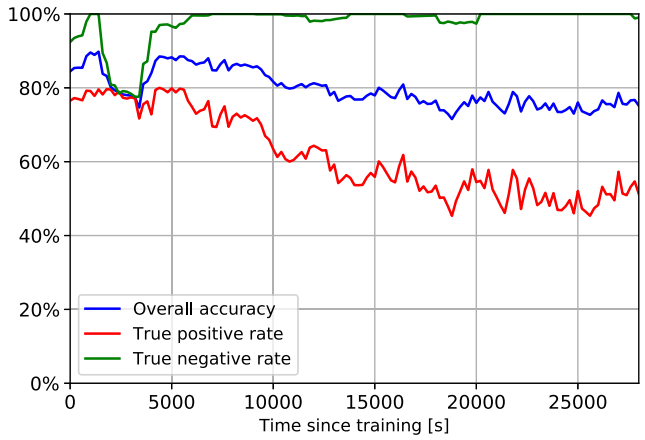

FIG. 8. Accuracy, true positive rate, and true negative rate over time during the O3 test period. For stability, these quantities are computed over rolling windows of length 2000 s beginning at the indicated time. Note that the scale of the x axis is different from Fig. 7.

more auxiliary channels and decoupling glitch identification from the GW strain, our EMU method provides important independent verification of glitches, as well as more information about the detector state.

C. Astrophysical implications

We illustrate the implications of these results in an astrophysical setting with the following examples:

In a use-case scenario where we aim to recover so-called *subthreshold* events that are *lost* by traditional GW search methods because they are not classified as detections, it is admissible to have a sizeable false dismissal rate of real signals, but it is desired to have a high glitch rejection rate. We find that typically a ∼65% glitch rejection rate (i.e., a 35% false negative rate) in individual detectors can be assumed at the cost of $\lesssim$0.3% false positives for the high-SNR case (see, e.g., high-SNR O3 curve in Fig. 6). The proposed method considers data from individual detectors independently, so with such a threshold the chance of a coincident false negative at all three sites is less than 5%, and the chance of a false positive at one or more sites is ∼1%. Therefore, a ∼95% reduction in triple-detector coincident glitches corresponds to a negligible chance [i.e., ∼O(1%)] to miss a true GW signal (i.e., the false positive rate of the glitch rejection is small).

Since in this scenario it is sufficient to flag any of the glitches contributing to the triple coincidence, the method can lead to approximately an order-of-magnitude reduction in glitch-dominated false alarm rates (FARs) for triple-detector events. Let us consider a fiducial subthreshold FAR of 10^{-7} Hz for a transient event candidate that is not sufficient for detection claims, and assume that the FAR is only determined by the triple-detector glitch rate from these glitches. If we decrease the triple-detector glitch rate by over an order of magnitude, then the hypothetical GW event candidate events moves to the detectable FAR region of 10^{-8} Hz.

In an another use case scenario, we can consider all transients detected. It is then imperative to have a very low false dismissal possibility for real signal, so we can choose a different strategy and operate at a different set point on the ROC curves displayed on Fig. 6. For example, considering that current observation runs produce ∼O(100) discoveries, one might require that the false dismissal probability to

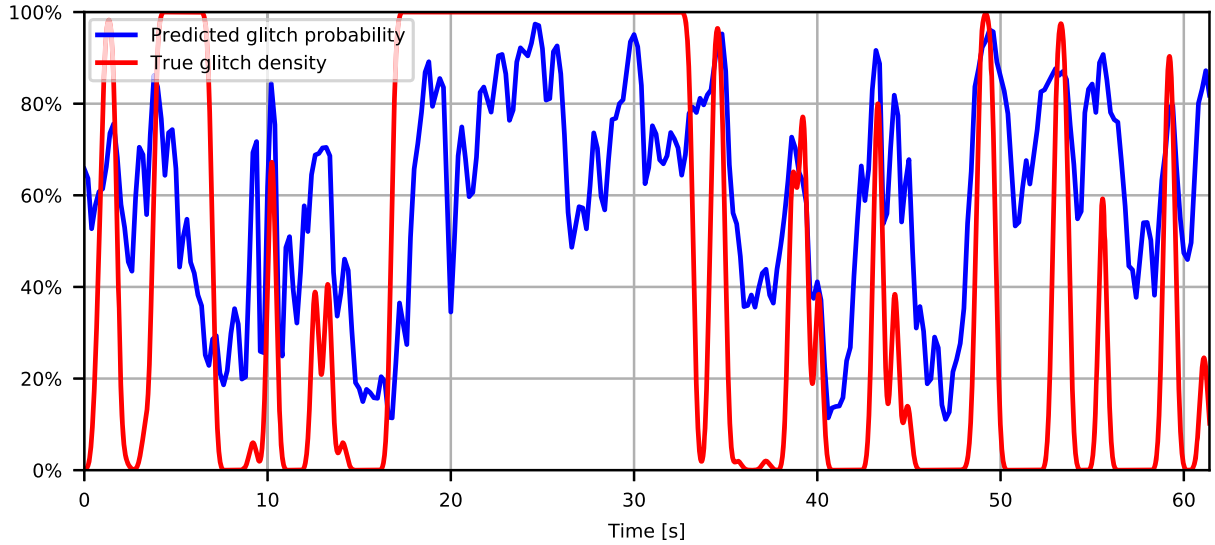

FIG. 9. Visualization of estimated glitch probability (blue) and true glitch density (red) over time during a 64-second segment from early O3. (Glitch density was calculated by checking whether Omicron reported a glitch of any SNR during each sample time, then smoothing the resulting binary vector by convolving it with a Gaussian with $\sigma = 0.2$ s. As a result, very short-duration glitches surrounded by longer glitch-free periods appear as low spikes of the red line—for example, around 9 s, the model correctly identifies the presence of a very short glitch, which is not obvious in the red line because of the glitch's short duration.)

accidentally miss a true GW signal be less than ∼O(0.1%). We can then require that all three detectors' glitches be flagged to have a triple coincident glitch flagged. Consequently, we need to operate at or below the ∼O(10%) false dismissal rate. This corresponds to ∼90% individual true positive rate on the ROC curves displayed on Fig. 6, resulting in a factor-of-several reduction in the triple-detector glitch rate.

These examples indicate some of the astrophysical opportunities presented by the results here.

D. Glitch subsets

For all of the results presented previously, we considered all triggers recovered by Omicron together without regard to any of their parameters (except for the high-SNR subsets illustrated in Fig. 6). However, the EMU method could also be used independently on subsets of glitches, which could be defined according to any desired criteria, such as frequency, duration, or other Omicron parameters; by suspected origin; or by using any of the existing methods that attempt to identify groups of related glitches [6]. One would expect that some groups of glitches might be easier to classify than others; for example, we noticed that glitches with higher peak frequency or longer duration were generally easier to classify than those with lower peak frequency or shorter duration.

To illustrate this, we performed an experiment in which, prior to training our model, we performed *k*-means clustering [35,36] on the (log) duration, peak frequency, bandwidth, and (log) SNR (each as reported by Omicron) of glitches in our ER14 training dataset. *k*-means is a standard clustering algorithm that attempts to identify clusters of related points in a provided dataset. It outputs the centroid of each cluster and the assignment of each data point to a cluster.

We divided the ER14 training dataset into 10 subsets (10 was chosen arbitrarily) identified by *k*-means and trained 10 elastic net logistic regression classifiers, using one of the glitch subsets as the positive class and all glitch-free samples as the negative class for each classifier. For testing, we then used the cluster centroids computed on the training set to assign each data point of the ER14 validation dataset to one of the 10 classes and evaluated the performance of each classifier on its corresponding validation subset. The results are shown in Fig. 10.

We note that the best performing cluster corresponded to glitches with relatively high bandwidth, signal-to-noise ratio, and peak frequency, and relatively long duration. While several works have attempted to identify auxiliary channels correlated with groups of glitches determined by various means [6,11], none of them has considered the full set of auxiliary channels; our EMU method would enable them to do so, agnostic to the grouping method used. We leave further exploration of this method and phenomenon to future work.

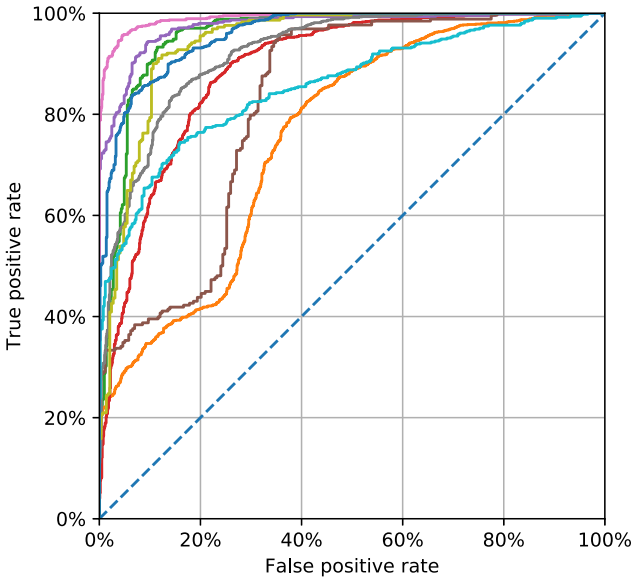

FIG. 10. ROC curves for 10 subsets of glitches from the ER14 dataset, pregrouped by *k*-means clustering on their Omicron parameters. We find best performance (the pink curve) on a cluster of high duration, bandwidth, signal-to-noise ratio, and peak frequency; the worse performing clusters (e.g., orange) have very short duration (order of milliseconds) and low SNR.

IV. CONCLUSIONS

We have presented a method for flagging potentially glitchy segments of LIGO data using only auxiliary channel data. It is the first method that considers *all* auxiliary channels to identify glitches without directly using the GW strain data. As such, it can provide independent corroboration that an event of interest is likely to be astrophysical in origin or not. It uses a well-established, easily interpretable, and efficient machine learning method.

We report a typical overall accuracy of approximately 80%, tested on segments from LLO during ER14 and O3, and find that the performance can be improved for certain subsets of glitches. We also show that this method is capable of reducing glitch-related false detections with negligible false dismissal for a detector network. The method also provides interpretable results, indicating specific auxiliary channel behavior associated with glitches and its predictions, which could facilitate detector improvements.

We note that the performance characteristics of the method are adjustable and can be modified to suit the requirements of a given study. Because the method outputs a probability estimate on a continuous scale for each time sample, one can vary the threshold used to classify a sample as glitchy or clean as necessary to achieve a desired balance between false positive and false negative rates.

The method can also be trained on a subset of glitches chosen based on characteristics such as SNR, frequency, or time duration—or any other desired criteria—to target its performance toward similar glitches. We can also specify a subset of the channels to consider in the feature set. This can

be used to focus on a specific detector subsystem or specific application. We leave the investigation of these applications to future work. We also note that the goal of identifying glitches using auxiliary channels is method independent, and other algorithms can be tested and compared to these results.

The classification model considers a set of features derived from the raw auxiliary channel data at and around a given time. The feature set was chosen in an *ad hoc* manner based on intuition, and we did not significantly attempt to engineer it for performance. We believe the simplicity of the features is an advantage considering the size of our dataset and illustrates the robustness of the machine learning model we employ. We note, however, that the model is agnostic to the features used—exactly the same type of model could be employed on top of more advanced, better optimized features. Considering the engineering of features more carefully represents a worthwhile direction for future work.

Finally, the method presented is not limited to GW detectors, and represents a general approach to analysis of the status of a complex system using large numbers of features as input. It could also be employed in studying other complex machines, experiments, or applications involving similarly high-dimensional data.

ACKNOWLEDGMENTS

We acknowledge computing resources from Columbia University's Shared Research Computing Facility project, which is supported by NIH Research Facility Improvement Grant No. 1G20RR030893-01, and associated funds from the New York State Empire State Development, Division of Science Technology and Innovation (NYSTAR) Contract No. C090171, both awarded 15 April 2010. The authors are grateful for the LIGO Scientific Collaboration review of the paper, and this paper is assigned a LIGO DCC number (P1900303). The authors acknowledge the LIGO Collaboration for the production of data used in this study and the LIGO Laboratory for enabling Omicron trigger generation on its computing resources (National Science Foundation Grants No. PHY-0757058 and No. PHY-0823459). The authors are grateful to the authors and maintainers of the Omicron and Omega pipelines, the LIGO Commissioning and Detector Characterization Teams and LSC expert colleagues whose fundamental work on the LIGO detectors enabled the data used in this paper. The authors would like to thank Stefan Countryman and William Tse for their help and suggestions, as well as colleagues of the LIGO Scientific Collaboration and the Virgo Collaboration for their help and useful comments, in particular Joe Betzweiser, Marco Cavaglià, Sheila Dwyer, Reed Essick, Patrick Godwin, and Jess McIver, which we hereby gratefully acknowledge. The authors thank the University of Florida and Columbia University in the City of New York for their generous support. The authors are grateful for the generous support of the National Science Foundation under Grant No. CCF-1740391.

[1] B. P. Abbott *et al.*, Classical Quantum Gravity **32**, 074001 (2015).
[2] F. Acernese *et al.*, Classical Quantum Gravity **32**, 024001 (2015).
[3] B. P. Abbott *et al.*, Classical Quantum Gravity **35**, 065010 (2018).
[4] J. Rollins, Ph.D. Thesis, Columbia University, 2011, https://gwic.ligo.org/assets/docs/theses/rollins_thesis.pdf.
[5] S. Chatterji, L. Blackburn, G. Martin, and E. Katsavounidis, Classical Quantum Gravity **21**, S1809 (2004).
[6] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. K. Katsaggelos, S. L. Larson, T. K. Lee, C. Lintott, T. B. Littenberg, A. Lundgren, C. Østerlund, J. R. Smith, L. Trouille, and V. Kalogera, Classical Quantum Gravity **34**, 064003 (2017).
[7] F. Robinet, Omicron: An algorithm to detect and characterize transient noise in gravitational-wave detectors, https://tds.ego-gw.it/ql/?c=10651, 2015.
[8] J. Areeda, J. Smith, A. Lundgren, E. Maros, D. Macleod, and J. Zweizig, Astron. Comput. **18**, 27 (2017).
[9] B. P. Abbott *et al.*, Classical Quantum Gravity **33**, 134001 (2016).
[10] T. Isogai (LIGO Scientific, and Virgo Collaborations), J. Phys. Conf. Ser. **243**, 012005 (2010).
[11] J. R. Smith, T. Abbott, E. Hirose, N. Leroy, D. MacLeod, J. McIver, P. Saulson, and P. Shawhan, Classical Quantum Gravity **28**, 235005 (2011).
[12] P. Ajith, T. Isogai, N. Christensen, R. X. Adhikari, A. B. Pearlman, A. Wein, A. J. Weinstein, and B. Yuan, Phys. Rev. D **89**, 122001 (2014).
[13] R. Essick, L. Blackburn, and E. Katsavounidis, Classical Quantum Gravity **30**, 155010 (2013).
[14] R. Biswas, L. Blackburn, J. Cao, R. Essick, K. A. Hodge, E. Katsavounidis, K. Kim, Y.-M. Kim, E.-O. Le Bigot, C.-H. Lee, J. J. Oh, S. H. Oh, E. J. Son, Y. Tao, R. Vaulin, and X. Wang, Phys. Rev. D **88**, 062003 (2013).
[15] R. Essick, Ph.D. Thesis, Massachusetts Institute of Technology, 2017, https://dspace.mit.edu/handle/1721.1/115024.
[16] M. Cavaglià, K. Staats, and T. Gill, Commun. Comput. Phys. **25**, 963 (2019).
[17] M. Walker, A. F. Agnew, J. Bidler, A. Lundgren, A. Macedo, D. Macleod, T. J. Massinger, O. Patane, and J. R. Smith, Classical Quantum Gravity **35**, 225002 (2018).

[18] M. Lightman, J. Thurakal, J. Dwyer, R. Grossman, P. Kalmus, L. Matone, J. Rollins, S. Zairis, and S. Márka, J. Phys. Conf. Ser. **32**, 58 (2006).
[19] J. Powell, D. Trifirò, E. Cuoco, I. S. Heng, and M. Cavaglià, Classical Quantum Gravity **32**, 215012 (2015).
[20] J. Powell, A. Torres-Forné, R. Lynch, D. Trifirò, E. Cuoco, M. Cavaglià, I. S. Heng, and J. A. Font, Classical Quantum Gravity **34**, 034002 (2017).
[21] N. Mukund, S. Abraham, S. Kandhasamy, S. Mitra, and N. S. Philip, Phys. Rev. D **95**, 104059 (2017).
[22] M. Razzano and E. Cuoco, Classical Quantum Gravity **35**, 095016 (2018).
[23] D. George, H. Shen, and E. A. Huerta, Phys. Rev. D **97**, 101501 (2018).
[24] D. George and E. A. Huerta, Phys. Rev. D **97**, 044039 (2018).
[25] S. J. Kapadia, T. Dent, and T. Dal Canton, Phys. Rev. D **96**, 104015 (2017).
[26] S. Vinciguerra, M. Drago, G. A. Prodi, S. Klimenko, C. Lazzaro, V. Necula, F. Salemi, V. Tiwari, M. C. Tringali, and G. Vedovato, Classical Quantum Gravity **34**, 094003 (2017).
[27] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).
[28] T. S. Adams, D. Meacher, J. Clark, P. J. Sutton, G. Jones, and A. Minot, Phys. Rev. D **88**, 062006 (2013).
[29] K. Kim, I. W. Harry, K. A. Hodge, Y.-M. Kim, C.-H. Lee, H. K. Lee, J. J. Oh, S. H. Oh, and E. J. Son, Classical Quantum Gravity **32**, 245002 (2015).
[30] P. T. Baker, S. Caudill, K. A. Hodge, D. Talukder, C. Capano, and N. J. Cornish, Phys. Rev. D **91**, 062004 (2015).
[31] C. Dreissigacker, R. Sharma, C. Messenger, R. Zhao, and R. Prix, Phys. Rev. D **100**, 044009 (2019).
[32] D. George and E. A. Huerta, Phys. Lett. B **778**, 64 (2018).
[33] A. J. K. Chua and M. Vallisneri, Phys. Rev. Lett. **124**, 041102 (2020).
[34] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, arXiv:1909.06296.
[35] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer-Verlag, Berlin, Heidelberg, 2006), https://www.springer.com/us/book/9780387310732.
[36] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction* 2nd ed. (Springer, New York, 2009), https://www.springer.com/us/book/9780387848570.
[37] H. Zou and T. Hastie, J. R. Stat. Soc. Ser. B **67**, 301 (2005).
[38] K. Blackburn, B. Mours, S. Anderson, A. Lazzarini, J. Zweizig, and E. Maros, Specification of a common data frame format for Interferometric Gravitational Wave Detectors (IGWD), CERN Tech. Report No. LIGO-T970130-v2, 2019, https://dcc.ligo.org/LIGO-T970130/public.
[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, J. Mach. Learn. Res. **12**, 2825 (2011), http://jmlr.org/papers/v12/pedregosa11a.html.
[40] R. Tibshirani, J. R. Stat. Soc. Ser. B **58**, 267 (1996).