Information Sciences 444 (2018) 172–186

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Machine learning for Gravity Spy: Glitch classification and dataset

S. Bahaadini[a,*], V. Noroozi[b], N. Rohani[a], S. Coughlin[c,d], M. Zevin[c,d], J.R. Smith[e], V. Kalogera[c,d], A. Katsaggelos[a]

[a] Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA
[b] Department of Computer Science, University of Illinois at Chicago, IL, USA
[c] Center for Interdisciplinary Exploration and Research in Astrophysics (CIERA), Northwestern University, Evanston, IL, USA
[d] Department of Physics and Astronomy, Northwestern University, Evanston, IL, USA
[e] Department of Physics, California State University Fullerton, Fullerton, CA, USA

ARTICLE INFO

Article history:
Received 23 December 2017
Revised 25 February 2018
Accepted 28 February 2018
Available online 6 March 2018

Keywords:
Deep learning
aLIGO
Dataset
Machine learning
Classification
Gravity Spy

ABSTRACT

The detection of gravitational waves with ground-based laser-interferometric detectors requires sensitivity to changes in distance much smaller than the diameter of atomic nuclei. Though sophisticated machinery and techniques have been developed over the past few decades to isolate such instruments from non-astrophysical noise, the detectors are still susceptible to instrumental and environmental noise transients known as "glitches," which hinder searches for transient gravitational waves. The *Gravity Spy* project is an effort to comprehensively classify the glitches that afflict gravitational wave detectors into morphological families by combining the strengths of machine learning algorithms and citizen scientists.

This paper presents the initial Gravity Spy dataset used for citizen scientist and machine learning classification – a static, accessible, documented dataset for testing machine learning supervised classification. Previous versions of this dataset used in [8, 53] did not include all current classes and also for some of the classes, some samples were pruned and added. This set consists of time–frequency images of LIGO glitches and their associated metadata. These glitches are organized by time–frequency morphology into 22 classes for which descriptions and representative images are presented. Results from the application of state-of-the-art supervised classification methods to this dataset are presented in order to provide baselines for future glitch classification work. Standard splitting for training, validation, and testing sets are also presented to facilitate the comparison between different machine learning methods. The baseline methods are selected from both traditional and more recent deep learning approaches. An ensemble framework is developed that demonstrates that combining various classifiers can yield a more accurate model for classification. The ensemble classifier, trained with the standard training set, achieves 98.21% accuracy on the standard test set.

* Corresponding author.
E-mail address: SaraBahaadiniBeigyZarandi2019@u.northwestern.edu (S. Bahaadini).

https://doi.org/10.1016/j.ins.2018.02.068
0020-0255/

1. Introduction

The recent observations of gravitational waves from binary black hole mergers [3–5,49] have inaugurated a new field of observational astronomy by providing a new method with which to explore the cosmos. These observations, made by the advanced Laser Interferometer Gravitational wave Observatory (LIGO, [28]), require sensitivity to fractional changes of distance on the order of 10^{-21}. The two LIGO detectors are in Hanford, Washington (LHO) and Livingston, Louisiana (LLO). To achieve this unprecedented sensitivity, all sensitive components of LIGO are exquisitely isolated from non-gravitational wave disturbances. Even with this isolation, the LIGO detectors are susceptible to disturbances that cause noise in the detectors and can afflict searches for gravitational waves.

Of particular concern are transient, non-Gaussian noise sources known colloquially as *glitches*. Glitches occur at a significant rate, come in many morphologies, and can mask or mimic gravitational wave signals. Work has been done in assessing whether or not an instance of excess noise is, in fact, a glitch [10], but a comprehensive classification and characterization of these noise features could allow their origin to be identified and their root cause to be removed from the instruments. Attempts to use machine learning algorithms have shown promise in glitch classification endeavors [35,36,40–42], however these techniques do not yet capture the full range of glitch morphologies present in LIGO data. In addition to the above methods, *Gravity Spy*[1] [53], a citizen science project hosted by the Zooniverse platform [12] that combines the classification power of machine learning and crowd-sourcing, provides a solution for addressing this problem. A critical component to the Gravity Spy method is the dataset used in training both the machine learning algorithm and the citizens.

In this paper, we present the Gravity Spy dataset, which is a collection of images of glitches and their associated metadata, in the context of machine learning tasks. We discuss the characteristics of the glitch classes within this data and provide an example for each class. To illustrate the complexity of the data and provide a better understanding of the relationship between various glitch classes, we visualize the feature space of Gravity Spy dataset. We further present a standard benchmark by defining the exact training, testing, and validation split sets that could be used to compare different machine learning algorithms. This dataset and standard benchmarks allow for further studies by the machine learning community, such as those performed in [21]. We apply state-of-the-art machine learning algorithms such as deep neural networks, support vector machines, and ensemble learning on this dataset to provide baselines for future works on this dataset. Although the problem considered in this paper is classification, the Gravity Spy dataset can be used for other machine learning tasks, such as clustering [51] and image retrieval [17].

Overall in this paper we have the following contributions:

- Introduction of the full specifications of the Gravity Spy dataset.
- Visualization of the Gravity Spy dataset.
- Determination of classification baseline accuracies for the dataset by developing classifiers based on neural networks, support vector machines, and ensemble learning, which are vital for establishing a control in testing future algorithms.

In the following sections, we describe the process of producing the images of glitches from raw gravitational wave detector data (Section 2.1), explain the specifications of glitch classes within this dataset (Section 2.2), investigate the feature space of all classes in the dataset (Section 2.3) and present the standard sets (Section 2.4). Finally, different machine learning baselines, with performance evaluation and data analysis for supervised classification tasks, are presented in Section 3. Concluding remarks are made in Section 4.

2. Gravity Spy data

Gravitational wave data, including transient noise in the detectors, is often visualized as time–frequency spectrograms. The images in the Gravity Spy dataset (used for both human classification and machine learning tasks) are a particular type of spectrogram based on decomposition using sine-Gaussian templates, a process known as the *Q-transform* [15]. The Gravity Spy dataset is composed of Q-transform images of any transients recorded by the gravitational wave channels of the detectors that exceed a certain threshold in loudness, specifically the signal-to-noise ratio (SNR), and pass the standard set of data quality criteria [2] used by LIGO's real-time gravitational wave searches.

Additionally, the Gravity Spy machine learning algorithms required a large training set of example images that belong to classes of morphologically distinct glitches to be constructed in order to allow machine learning pre-classification of the images that would be presented to citizen scientists [53]. To accomplish this, 22 different morphologically distinct classes of glitches were selected (the names and morphology of many of these classes had already been identified by the broader LIGO Scientific Collaboration [1,2,39]) and tens to hundreds of example images were hand selected (often with input from algorithms, such as the Hierachical Veto [48] that identify classes of glitches by their relationship with other types of disturbances, such as seismic noise). These classes are also the classification choices (buttons) that citizen scientists have to choose from the Gravity Spy project interface.

Over time, these training sets have expanded. When both the machine learning and volunteer classification of a given unlabeled image passes a certain confidence threshold [53], these images are "retired" and added to the training set. Furthermore, new glitch classes identified by citizen scientists, volunteers, or clustering algorithms are manually (following

[1] https://www.gravityspy.org.

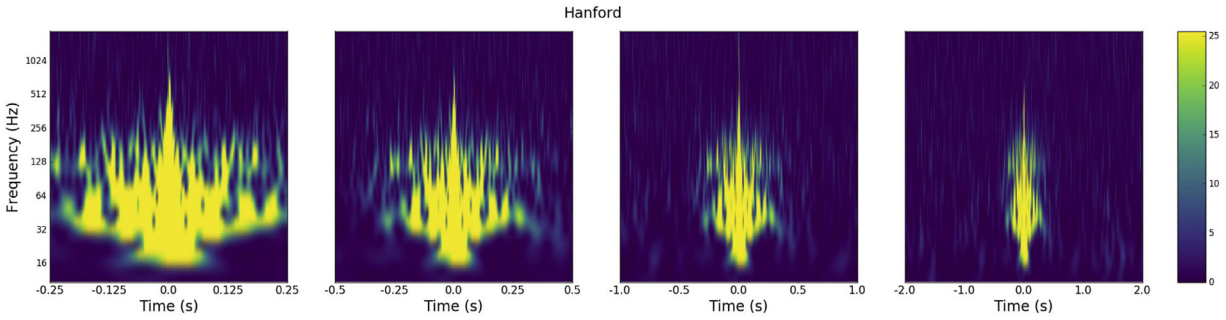


Fig. 1. Omega Scan images, with four different durations, of a glitch in the Koi Fish class. Color shows the 'loudness' of the signal.

discussion by the science team) added to the training set. More detailed information about this process can be found in [53]. Because the training set is continually evolving, this paper presents only the initial training dataset which does not include the so called "retired" images for the purposes of creating a static reference for other groups interested in glitch classification. The dataset used in [8,53] is a smaller version of the current dataset with 20 classes. Furthermore, we prune the samples of [8,53] for the current dataset and add new examples to some of the classes.

2.1. Data preparation

The Gravity Spy data set is constructed using the output of a transient excess power identification algorithm, Omicron [30,43]. Omicron, which itself is based on the Q-transform, uses sine-Gaussian wavelets to identify transient excess noise in the data and clustering to identify metadata features about this excess noise such as peak frequency and SNR. In order to focus on the excess noise that is most troubling to gravitational wave searches, the following filters are applied to the glitch triggers. First, glitches that occurred outside of time periods when their detector was in "observing mode" are rejected. Observing mode indicates that the configuration and state of the detector is nominal and the data is ready to be searched for gravitational waves [2]. Second, glitches with SNR (as reported by Omicron) below 7.5 are rejected, since glitches below this threshold are difficult to see in the images and thus difficult to classify based on their morphology. Third, glitches whose peak frequency falls outside of LIGO's most sensitive frequency band, from 10 Hz to 2048 Hz, are also rejected.

After applying these filters, time–frequency representations, called Omega Scans [15] and again based on the Q-transform, of the raw data for the remaining glitches are created and saved as image files. Omega Scans [15] are outstanding at visualizing a large range of transient noise that have a similar time–frequency morphology. Omega Scans represent a generic signal as a combination of sine-Gaussians. Omega Scans perform an unmodeled SNR calculation with the template for a signal defined by its 'Q' value, where Q is the quality factor of a sine-Gaussian waveform. An Omega Scan searches over a range of time-frequency tilings constructed using different Q templates and identifies the template that gives the loudest SNR value, then generates a spectrogram image of the time–frequency tiling corresponding to that Q. The color axis of this image is the normalized energy (roughly SNR squared), defined as the square of a given tile's Q transform magnitude divided by the mean squared magnitude in the presence of stationary white noise:

$$Z = \frac{|X|^2}{\langle |X|^2 \rangle} \tag{1}$$

where Z is the normalized energy and $|X|$ is the Q transform magnitude of a tile [15].

Fig. 1 shows example Omega Scan images for a glitch from the Koi Fish class (described below) that occured at LHO. Each Omega Scan image is centered on the time of maximum energy of the glitch, and each glitch is visualized using four different time windows ($\pm$ 0.25, 0.5, 1.0, and 2.0 s, referred to as view 1, view 2, view 3, and view 4, respectively, in the rest of this paper) to accommodate the durations that are most common for the different glitch classes. For the classification of a given glitch, both citizen scientists and machine learning algorithms are presented images of all four time durations. For more details about views, i.e., time durations, see [8].

2.2. Dataset specifications

Below are listed the 22 classes of glitches included in this dataset. Examples from each class are shown in Fig. 2. The names associated with these classes and the typical morphology of the glitches belonging to each class were set by a combination of LIGO scientists, Gravity Spy scientists, and in some cases, citizen scientists. Many of the glitch classes listed here, and in some cases their physical causes, were previously identified in work to characterize the LIGO data [1,2,39]. Glitch classes for which LIGO scientists have uncovered or fixed the cause are also highlighted below.

These classes are not exhaustive and they are not static. Because environmental conditions at the sites change with weather and seasons and because the LIGO detectors are under active commissioning to bring them to their designed levels

of sensitivity and robustness, the classes of glitches change with time. There are many possible sub-classes of glitches and there are sometimes new or short-lived classes of glitches. These 22 classes are an attempt at delineating the most representative and distinguishable classes during LIGO's first and second observing runs, from September 2015 to December 2015 (O1) and November 2016 to August 2017 (O2), respecitvely.

1. *1080 Lines* At LIGO Hanford during O2, there was a steady stream of glitches around 1080 Hz. In Omega Scan images these appeared as a string of yellow dots, sometimes connected to form a line, and sometimes more sparse. These glitches were greatly reduced following a configuration change that increased the gain of a control loop for the LHO output mode cleaner's length (see LHO electronic logbook entry 33,104 [18]). 1080 Hz lines were most prevalent between 11 October 2016 (i.e., during the engineering run before the start of O2) and 14 January 2017.
2. *1400 Ripples* These are somewhat strong short-duration glitches at around 1400 Hz. They can appear isolated (one glitch per image) though sometimes multiple glitches are seen within a 4-s-long image. So far, the source of these glitches is unknown. This class was inspired by collections of glitches and forum discussions by Gravity Spy citizen scientists.
3. *Air Compressor* These short-duration glitches look like a thick line centered at a frequency of 50 Hz. At LIGO Hanford, these were found to be related to air compressor motors switching on and off at the end stations. This issue was solved on September 29, 2016 by replacing the vibration isolators (rubber feet) on the air compressors (see LHO electronic logbook entries 22,081 [20] and 21,436 [47]).
4. *Blip* These short glitches usually have a duration of around 40 ms, frequencies between 30 and 500 Hz, and typically appear as a narrow, vertical and symmetric 'teardrop' shape in time-frequency domain [2]. They appear in both Hanford and Livingston detectors and their root cause is unknown.
5. *Chirp* A "chirp" is the characteristic time-frequency shape created by gravitational waves from inspiraling compact objects, sweeping upwards in frequency over time. The only chirps in this Gravity Spy dataset are so-called "hardware injections" [11], simulated gravitational wave signals physically added to the detectors for testing and calibration purposes. Additionally, though it was not included in this dataset, Gravity Spy users were recently presented with images of the gravitational wave signal GW170104 [5].
6. *Extremely Loud* This is a catch-all category for glitches that result from a major disturbance in the detectors. They usually span much of the spectrogram and have extremely high energies. High energy glitches from other categories (e.g., Koi Fish) that saturate most of the image are also placed in this category.
7. *Helix* This category, inspired by the collections and forum comments made by Gravity Spy citizen scientists, contains glitches that resemble a vortex, occur at intermediate frequencies, and often come in groups. Their origin is unknown.
8. *Koi Fish* Koi Fish glitches are similar to Blip glitches, but they resemble a fish with the head at the low frequency end of the plot, pectoral fins around 30 Hz, and a thin tail around 500 Hz.
9. *Light Modulation* The morphology of this glitch is not always the same, but it often looks like several bright spikes in close succession, often with low frequency noise at the same time. They are caused by amplitude fluctuations in the signal used to generate the 45 MHz optical sidebands (used to control the length and alignment of some of LIGO's optical cavities).
10. *Low Frequency Burst* This is a catch-all category for loud, short-lived, low-frequency noise. Though multiple morphologies make up this category, they often resemble small humps with a nearly triangular shape growing from low frequency to a peak and then dying back down in a second or two. This glitch class was common in LIGO Livingston's first observing run.
11. *Low Frequency Line* This category appears as horizontal lines at low frequencies. They are distinct from Low Frequency Bursts because they are more long-lived and from Scattered Light (see below) because they do not vary noticeably in frequency over long durations.
12. *No Glitch* The No Glitch category is designated for spectrograms that have no apparent or obvious transient noise structure visible in the Omega Scan. The lack of visbility could be due to a number of factors including issues displaying glitches whose peak frequency is high (2 kHz) and duration is short. Because Omega scans implement what amounts to a log tiling (and the images are on a log scale) these glitches could be visibly shrunk to the point that they blend into the background.
13. *Paired Doves* This category, inspired by the collections and forum comments made by Gravity Spy citizen scientists, is a collection of repeating glitches that look similar to chirps at low frequencies, and alternate between increasing and decreasing frequency. These glitches are believed to be related to periods of excess 0.4 Hz motion of the beamsplitter at LIGO Hanford (see LHO electronic logbook entry 27,138 [29]).
14. *Power Line* Power Line glitches usually look narrow in frequency, and last for about 0.2–0.5 s in time, centered around 60 Hz or one of its harmonics. They are due to glitches in the United States mains power (alternating current at a frequency of 60 Hz).
15. *Repeating Blips* This category is for blip glitches that repeat in the Gravity Spy images. Though this category encompasses all cadences of repetition, they are often found to repeat every 0.25 or 0.5 s.
16. *Scattered Light* Scattered light glitches come in many different morphologies, though they are often low-frequency, long duration, humpy glitches that look like one or several curved lines stacked on top of each other. They are due to light from the main LIGO laser beam path being scattered off moving objects and re-combining with the main beam but with a rapidly varying phase [6].

17. *Scratchy* This is a series of short-duration repeating glitches (often with 10–30 glitches per second) with intermediate frequencies, often lasting several seconds. They occur mostly at LIGO Hanford and some of them are related to scattered light from a "baffle" used to intercept stray laser light. This was determined due to the change in the characteristic of the glitch after damping was done to this "baffle" [45]. Recognizing them with usual machine learning classification methods is non-trivial because their morphologies can be scattered through a wide range of feature space. The name Scratchy comes from how this type of glitch sounds when the signal is converted to an audio signal.
18. *Tomte* These glitches are similar to, or possibly a sub-class of blips. They are lower-frequency glitches that are usually triangular in shape, and look like the hat worn by a garden gnome.
19. *Violin Mode Harmonic* This category appears short and dot-like, and occurs at 500 Hz and multiples (harmonics) of this frequency. LIGO's main mirrors are suspended by thin glass fibers that have resonances like that of a violin string. These glitches occur at the frequencies of those resonances.
20. *Wandering Line* These are lines that last on the order of minutes to an hour and meander in frequency. Some example causes of wandering lines include motors (such as vacuum pumps) that do not have fixed frequencies and beats between higher frequency signals that have some frequency variation.
21. *Whistle* Whistles have a characteristic "W" or "V" shape that sweeps down to lower frequencies. They are caused by radio frequency signals (i.e. signals at MHz frequencies) that interfere and beat with the LIGO Voltage Controlled Oscillators. The name Whistle comes from how this type of glitch sounds when the signal is converted to audio.
22. *None of the Above* This category is a catch-all for glitches that do not fit into the other 21 categories, and therefore has much variability in its morphology.

2.3. Gravity Spy dataset feature space

To better understand the distribution samples among various classes in Gravity Spy dataset, particularly the (dis)similarity between members of these glitch classes, we visualize the glitch images in the raw pixel feature space. The original glitch images are 570×470 pixels. We downsample them (to reduce the computation cost) to 140×170 and then employ the t-distributed Stochastic Neighbor Embedding (t-SNE) method [31] to project them into a two dimension feature space (see Fig. 3).

t-SNE is a nonlinear method for dimensionality reduction to visualize high-dimensional data. It projects samples in the original high-dimensional feature space (original dataset) $\mathcal{X} = \{x_1, \ldots, x_N\}$ into a new lower dimensional (usually two) feature space $\mathcal{Y} = \{y_1, \ldots, y_N\}$ such that the similarities between points would be preserved. This means that points that are close in the original high-dimensional feature space should remain close in the low-dimensional feature space and dissimilar points in the original space should be far from each other in the projected space. t-SNE estimates the similarity of x_j to x_i by the conditional probability of $p_{j|i}$ that point x_i selects point x_j as its neighbor as

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \tag{2}$$

where σ_i is the variance of the Gaussian centered on x_i [31]. The probability p_{ij} is proportional to the similarity of two points x_i and x_j and is estimated by $p_{ij} = \frac{p_{i|j}+p_{j|i}}{2N}$ where N denotes the total number of points. The similarities between the projected points q_{ij} are estimated as

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq m}(1 + \|y_k - y_m\|^2)^{-1}} \tag{3}$$

To obtain the location of the samples in the low dimensional feature space, t-SNE minimizes the KL divergence between p_{ij} and q_{ij} (due to their probabilistic nature) as

$$\text{t-SNE objective} = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{4}$$

Gradient descent is used to optimize this objective function.

Examining this low dimensional representation obtained by t-SNE, we observe that some classes are well separated in this space while others overlap. For instance, members of the Extremely Loud class are separated into three distinct groups, one of which overlaps with the Koi Fish class (these are the extremely loud Koi Fish). Similarly, there is a large cluster of Blip glitches that are mixed with Repeating Blips which are known to be morphologically similar. We also observe that few samples from the Koi Fish class are close to the Blip class which makes sense for the less loud Koi Fish glitches, which look similar to Blips. Scattered Light and Low Frequency Line also have a small overlap which is again consistent with citizen scientists' experience with these classes. Other points of Scattered Light are well clustered on the top right of the plot. In general, None of The Above glitches are spread all over the space and overlap with other classes. This makes sense because this catch-all group can have varied morphology. However, that also makes the classification of glitches into this class challenging and highlights the need for creating more classes that better capture the glitches currently put into None of The Above. These results agree with prior information about the morphological characteristics of these classes.

Table 1
Specifications of Gravity Spy dataset classes. The total number of examples for each class is shown in the Total column. The Duration column indicates whether the class has a short or long duration. The Frequency column shows the morphological structure of the class that may happen at low, mid or high frequency. The morphological structure may evolve over time or not which is reported in Evolving column for each class.

Class	Total	# train set	# valid set	# test set	Duration	Frequency	Evolving
1080Lines	328	230	49	49	Long	High	No
1400Ripples	232	162	35	35	Short	High	No
Air Compressor	58	41	8	9	Short	Low	No
Blip	1869	1308	281	280	Short	Mid	Yes
Chirp	66	46	10	10	Short	Mid, Low	Yes
Extremely Loud	454	318	68	68	Long	High, Mid, Low	Yes
Helix	279	195	42	42	Short	Mid	Yes
Koi Fish	830	581	125	124	Short	Mid, Low	Yes
Light Modulation	573	401	86	86	Long	Mid, Low	Yes
Low Frequency Burst	657	460	99	98	Short	Low	Yes
Low Frequency Lines	453	317	68	68	Long	Low	No
No Glitch	181	127	27	27	Long	–	No
None of the Above	88	62	13	13	Short	High, Mid, Low	Yes
Paired Doves	27	19	4	4	Short	Mid, Low	Yes
Power Line	453	317	68	68	Short	Low	No
Repeating Blips	285	200	69	42	Short	Mid	No
Scattered Light	459	321	69	69	Long	Low	Yes
Scratchy	354	248	53	53	Long	High, Mid	Yes
Tomte	116	81	17	18	Short	Low	Yes
Violin Mode	472	330	71	71	Short	High	No
Wandering Line	44	31	6	7	Long	High	Yes
Whistle	305	213	46	46	Short	High	Yes

Near the top of the plot, a cluster of Tomte glitches is very near to the None of The Above glitches. This lack of separation needs further investigation. On the mid left of the plot, we observe that Light Modulation glitches and Low Frequency Burst glitches are close to each other and have some overlap, likely due to the fact that Light Modulation glitches are often accompanied by low-frequency glitches (indeed our citizen scientists often mix classifications between these two classes). On mid bottom, there is a small green cluster (Air Compressor) which has some pink '+' samples (Power Line), these glitches do look morphologically similar; however the former occurs usually at 50 Hz while the later occurs at 60 Hz. The most dense area of overlap is in the low-mid center of the plot where Whistle, Violin Mode, 1080Lines, and Wandering Line glitches are found. These are all relatively high-frequency effects, but should have morphological distinctions, however especially for whistles while the shape of each glitch usually looks like a "V" or a "W" these may be narrow or wide, and thus may resemble these other categories. Overlaps between these classes may be improved in the future through better class definitions and improvements to the glitch classification training set.

2.4. Dataset version and standard split

The dataset of Gravity Spy is evolving, however here we describe only the initial dataset, named GravitySpyVersion1.0. It has a total of 8583 glitch samples from 22 classes (see Table 1). The dataset is split into 6008 train, 1288 validation, and 1287 test samples. As the distributions of the samples in the various classes are highly skewed, we force all classes to be populated in all training, validation, and test sets proportional to their distribution in the whole dataset.

Following the standard split of GravitySpyVersion1.0, the training set is used for estimating the model's parameters. The validation set is only used for tuning the model's hyper-parameters. For the sake of fair comparison to the presented baselines, the validation set is not added to the training set for training the model parameters. The test set is the unseen part of the dataset that can only be used for the final performance evaluation.

3. Classification methods

In this section, we first describe the application of three classification methods on the GravitySpyVersion1.0 dataset. The methods employed here are well-known and state-of-the-art from two main categories of machine learning algorithms: shallow models, such as logistic regression and support vector machines, which can be considered as traditional approaches in machine learning and the more recent deep neural network models [27,38,54]. We then describe the application of what we refer to as the Ultimate Ensemble method, to the GravitySpyVersion1.0 dataset, which consists of the combination of a number of classifiers.

3.1. Logistic regression

Logistic regression is a discriminative linear classifier that finds a hyperplane to separate the samples [51]. We denote the training set of N labeled examples as $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, with x_i representing the input samples and $y_i \in \{1, \ldots, C\}$

the corresponding labels, where C is the total number of glitch classes. We use the multinomial logistic regression which is the extension of the standard binary logistic regression to multi-class scenarios [25,51]. Multinomial logistic regression minimizes the following objective function with respect to the unknown hyperplane parameters θ^c per class c

$$\mathcal{J}(\theta) = \sum_{i=1}^{N}\sum_{c=1}^{C} 1\{y_i = c\} \log \frac{\exp\left((\theta^c)^T x_i\right)}{\sum_{c=1}^{C} \exp^{((\theta^c)^T x_i)}} \tag{5}$$

where $1\{a = b\}$ is the indicator function that returns one if the two input arguments are equal and zero otherwise.

The model is trained using the training set and evaluated over the test set. As logistic regression does not have any hyper-parameters, we do not use the validation set here. We use the scikit-learn [13] implementation of multinomial logistic regression with newton-cg as the solver.

The original resolution of the Gravity Spy spectrogram is 470×570. As a first step we investigated the effect of the resolution of the spectrograms on the performance of the classifier. If a lower resolution were to perform better (or at least as good as a higher resolution) then a double benefit would be obtained of higher performance and a lower computation cost.

The performance of multinomial logistic regression for different resolutions and different views is shown in Fig. 4. The original glitch images are down-sampled by a factor of 0.1, 0.2, 0.3, 0.4 and 0.6 resulting in glitch images of size 47×57, 93×113, 140×170, 186×226, and 282×342 pixels, respectively. Downsampling was performed with the 'rescale' function and the glitches were converted to gray-scale with the 'rgb2gray' function from Skimage toolbox. As can be observed from Fig. 4, the classification accuracy initially improves by increasing the resolution but after a point decreases (exclusively view 1 which performs poorly). Considering also the performance of the classifier across views, we observe that the performance of view 1 (shortest duration) is poor, while the performance for view 4 (longest duration) is inconsistent across resolutions and lower than views 2 and 3 for all other than one resolution. Based on the specification of different glitches provided in Table 1, some glitches have short duration and others have long duration. Therefore, using only view 1, may not provide enough information regarding long duration glitches. Similarly, using only view 4 may not provide enough resolution for short duration glitches. Based on these observations we adopt resolution 140×170 pixels for the remaining of the paper as it seems that this resolution does not sacrifice accuracy for the computation cost.

The confusion matrix obtained from view 3 (2 s duration) along with the precision and recall are shown in Fig. 5. As can be seen the model can learn well some of the classes such as 1080 Lines, Blip, Chirp, while none of the above, paired doves and wandering line are more challenging. The advantage of logistic regression is that it is a simple and a computationally fast model, however is cannot necessarily capture the existing nonlinearities in the data.

3.2. Support vector machine (SVM)

SVM was proposed in 1992 and it quickly became the state-of-the-art of machine learning methods for classification for many years [33]. These days, although deep learning approaches outperform SVM in many applications, SVM is still competitive in some cases, especially for small datasets.

This classifier finds the optimal set of hyperplanes with the largest minimum distance between classes. Two types of SVMs, e.g., linear and kernel, are used here.

3.2.1. Linear SVM

Having a training set of N samples $\{x_i\}_{i=1}^{N}$ and training labels $y \in \{1, -1\}$, SVM solves the following optimization problem

$$\begin{aligned} &\min_{w,b} \frac{1}{2} \parallel w \parallel^2 + C\sum_i \zeta_i \\ &\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 - \zeta_i \quad \text{and} \quad \zeta_i \geq 0, \quad \text{for} i = 1, \ldots, N \end{aligned} \tag{6}$$

where C is the capacity constant, w and b are the hyperplane parameters (w is the vector of coefficients and b is a constant), and ζ is a slack variable for handling non-separable inputs and allowing approximate solutions when there is not a feasible one. For a small value of C, we do not penalize slack variables ζ, and as we increase C, we obtain a large margin for SVM. C is a parameter which controls the trade-off between penalizing the slack variables ζ_i and a large margin for the classifier [51]. The model's hyper-parameters are tuned by grid search and n-fold cross validation. After finding the optimal hyper-parameters, the classifier is trained by using just the training set and the classification performance on the test set is reported.

We use the scikit-learn [13] implementation of linear SVM, "LinearSVC" with internal implementation of liblinear [19]. LinearSVC uses one-vs-the-rest scheme in which M classifiers are trained, where M is the number of classes. The ith SVM is trained with the examples in the ith class having positive labels, and the examples from all other classes having negative labels. For each test sample, having the decision values from all M classifiers, the resulting label would correspond to the class which has the largest decision function value.

The classification accuracies of linear SVM trained and tested on images in different views are reported in Table 2. As can be seen, linear SVM performs the best when applied to view 2 and the worst when applied to view 4 which is inline with our prior investigation for logistic regression and deep models [8].

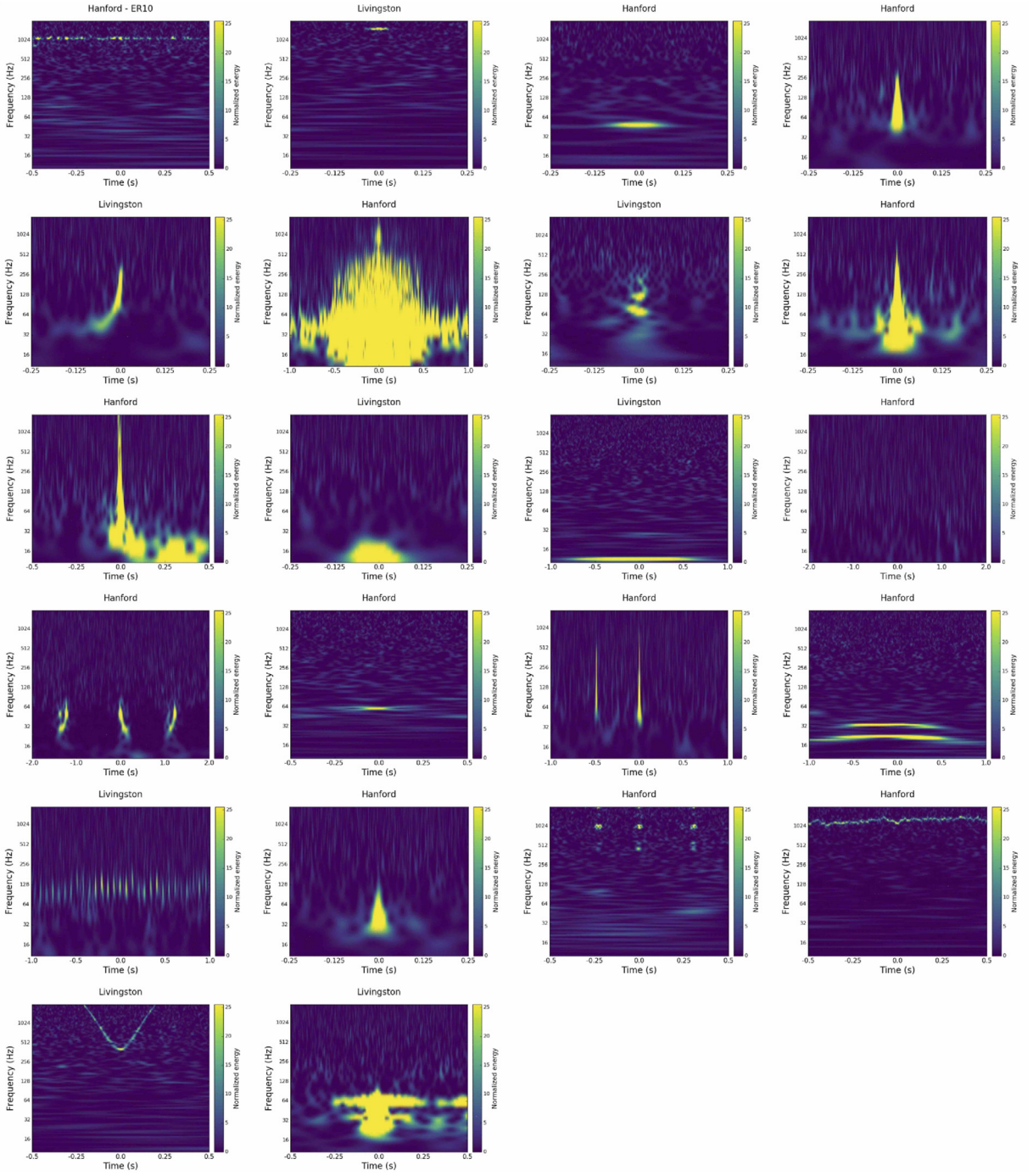


Fig. 2. Omega Scan images for example members of each class within the Gravity Spy dataset. From top left to bottom right; row one: 1080Lines, 1400Ripples, Air Compressor, Blip, row two: Chirp, Extremely Loud, Helix, Koi Fish, row three: Light Modulation, Low Frequency Burst, Low Frequency Lines, No Glitch, row four: Paired Doves, Power Line, Repeating Blips, Scattered Light, row five: Scratchy, Tomte, Violin Mode, Wandering Line, row six: Whistle, None of the Above (one possible example, this class can have various forms).

Table 2
Overall accuracy of linear SVM. Parameter $C = 0.1$ (obtained by grid search and n-fold cross validation).

View 1 (0.5 s)	View 2 (1 s)	View 3 (2 s)	View 4 (4 s)
93.93	**96.19**	95.88	93.16

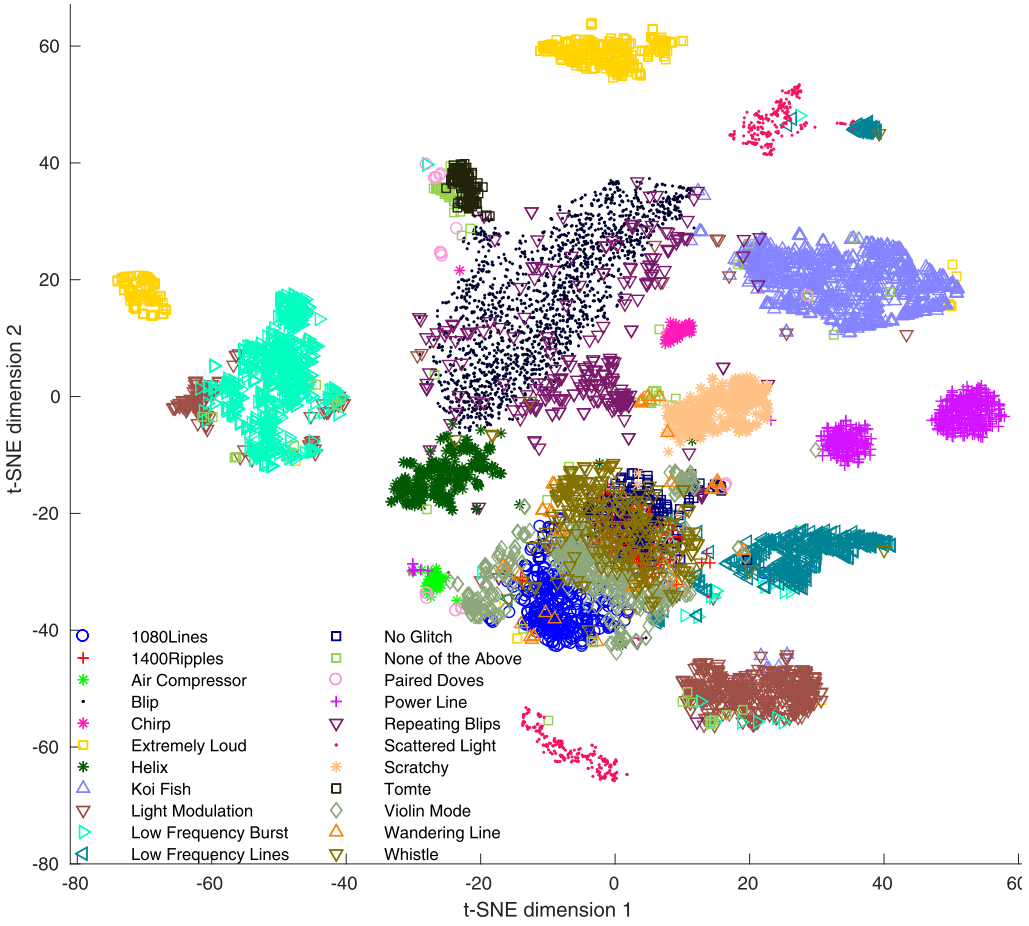


Fig. 3. Visualization of the Gravity Spy dataset feature space in two dimensions using the t-SNE [31] method. This 2-dimensional feature space represents separation in the original $140 \times 170 = 23,800$ dimensional space and is thus useful for understanding the distribution and overlap of glitches of various classes. This figure was produced using the one second duration images.

3.2.2. Kernel SVM

In many cases, the complexity present in the data cannot be captured by linear models. The kernel trick is applied in such scenarios as it projects the samples into another feature space where they can be linearly separated [7,14,51]. We use SVM the with the Radial Basis Function (RBF) kernel to capture the nonlinearity of Gravity Spy samples. Kernel SVM optimizes the following objective function

$$\begin{aligned} &\min_{w,b} \frac{1}{2} \| w \|^2 + C \sum_i \zeta_i \\ &\text{s.t.} \quad y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i \quad \text{and} \quad \zeta_i \geq 0 \quad \text{for } i = 1, \ldots, N \end{aligned} \tag{7}$$

The function $\phi(x_i)$ does not need to be explicitly defined. Instead, only the kernel $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ needs to be defined. The RBF kernel function is given by $K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|_2^2)$. The hyper-parameter γ controls the trade-off between error due to bias and variance in the model. A very large value of γ can lead to a very complex boundary with low bias/high variance and also having an overfitting problem. On the other hand a very small value of γ has the risk of a model with high bias/low variance. Here, for multi-class scenario, we use the one-vs-one scheme. This method constructs $\frac{M \times (M-1)}{2}$ classifiers, where M is the number of classes. Each classifier is trained on a pair of two classes. For a given test point, each classifier gives a vote and the final label is assigned to the class with the largest number of votes. The hyper-parameters of the RBF classifier, C and γ, are obtained using grid search and n-fold cross validation.

The classification accuracies of kernel SVM trained and tested on images in different views are reported in Table 3. The optimal parameters are found to be $C = 5.65$ and $\gamma = 4e{-}5$. As can be observed from Table 3, kernel SVM has the best performance when applied to view 2 and has the worst performance when applied to view 4.

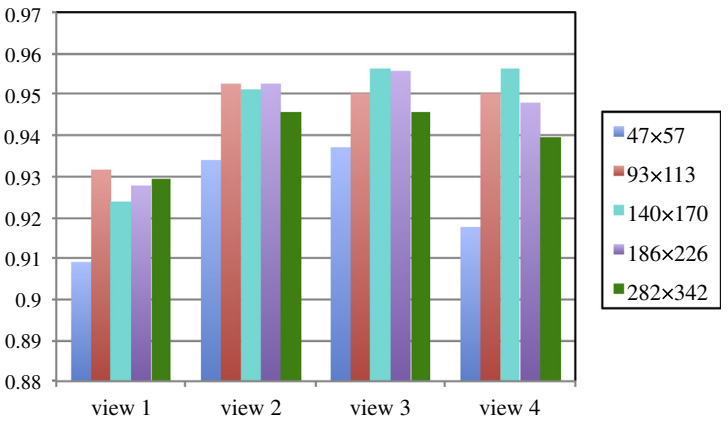


Fig. 4. Overall classification accuracy as a function of different views for various glitch image resolutions for multinomial logistic regression.

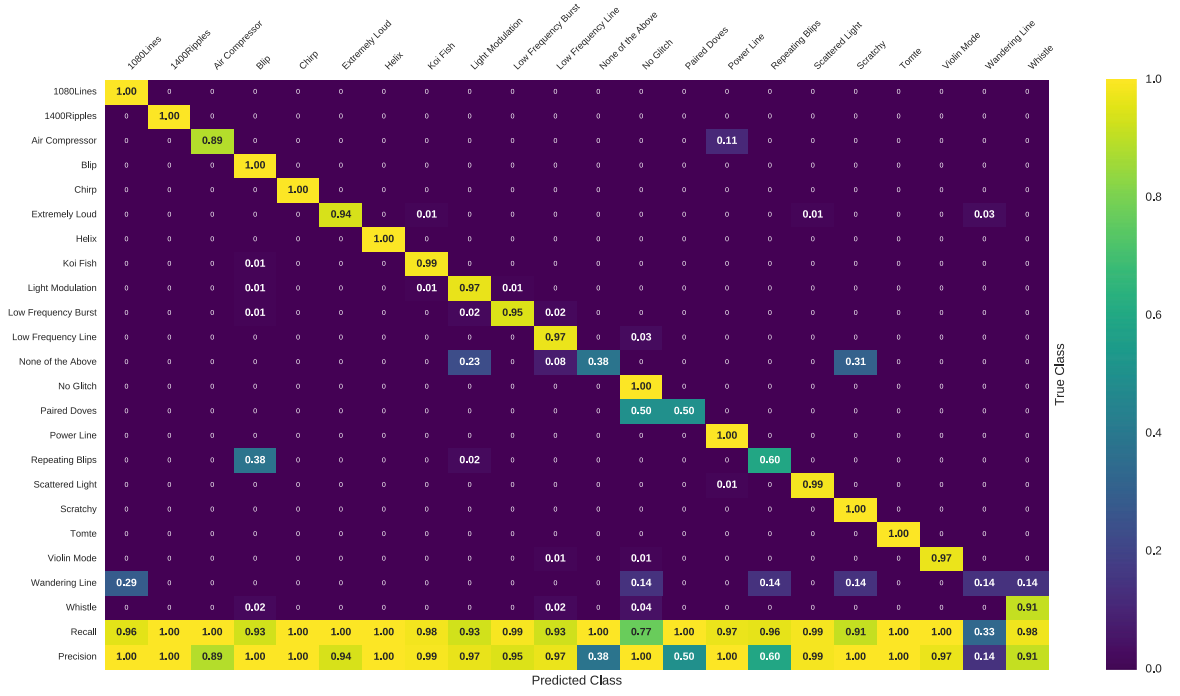


Fig. 5. Confusion matrix of logistic regression classifier applied to view 3 (2 s duration). Recall and precision values appended below for reference. The confusion matrix is normalized by the total number of glitches in each class so that values of precision and recall close to one show a more accurate classification.

Table 3
Overall accuracy of RBF kernel on different views. The RBF kernel parameters are $C = 5.65$ and $\gamma = 4e{-}5$.

View 1 (0.5 s)	View 2 (1 s)	View 3 (2 s)	View 4 (4 s)
95.41	**97.12**	96.50	93.93

3.3. Deep neural networks (DNNs)

Motivated by the impressive performance of DNNs for solving many classification problems [27], we choose them as one of the baselines for glitch classification for the Gravity Spy dataset.

Deep Convolutional Neural Networks (CNNs) [26] have shown superior performance on image data [24]. When CNNs are used for classification, they are usually composed of convolutional and max-pooling layers, followed by fully connected and softmax layers. A convolutional layer has a number of filters, which when applied to the input images produce the so

Table 4
Specifications for single and multi-view CNN used for glitch classification. We use the abbreviation of 'Reg' for regularizer. The number in parenthesis in front of convolutional and fully connected layer show the number of kernels and nodes, respectively.

Single view model	Merged-view model
Input 140×170	Input 280×340
5×5 Convolutional layer (32) with reg.	5×5 Convolutional layer (16) with reg.
2×2 Maxpooling, 0.5 drop-out	2×2 Maxpooling, 0.5 drop-out
5×5 Convolutional layer (64) with reg.	5×5 Convolutional layer (32) with reg.
2×2 Maxpooling, 0.5 drop-out	2×2 Maxpooling, 0.5 drop-out
Fully connected (256), 0.5 drop-out	5×5 Convolutional layer (64) with reg.
Softmax (22)	2×2 Maxpooling, 0.5 drop-out
	5×5 Convolutional layer (64) with reg.
	2×2 Maxpooling, 0.5 drop-out
	Fully connected (256), 0.5 drop-out
	Softmax (22)

Table 5
Accuracy of CNN on different glitch durations and merged-view model in [8] with new architecture.

View 1 (0.5 s)	View 2 (1 s)	View 3 (2 s)	View 4 (4 s)	Merged-view model
95.10	96.81	96.58	95.65	**97.67**

called feature maps, which are usually subsampled using the max (or mean) operation. Each node in a fully connected (FC) layer is connected to all the previous layer's nodes. Softmax is a FC layer popular for multi-class classification problems. The softmax layer size is equal to the number of the target classes. The output of the softmax layer is given by

$$o_c^i = \frac{e^{w_c^T x}}{\sum_{c=1}^{C} e^{w_c^T x}} \qquad \text{for } c = 1, \ldots, C \tag{8}$$

where o_c^i denotes the cth class score for the ith image as input to the model, x is the input given to the softmax layer (the output of the layer before softmax), w_c is the weight vector connecting to the cth node in the softmax layer and C is the total number of classes. We use cross-entropy as the objective function given by

$$\text{objective function} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_c^i \log o_c^i \tag{9}$$

where y_c^i denotes the binary label for sample i (it is one only when the sample i belongs to class c). There exits many optimization techniques [23,32,37,46,52]. We use Adadelta [52] to optimize the objective function.

We employ CNN for all four durations. The specifications of the CNN are reported in the left column of Table 4. The architecture of CNNs is optimized for the best classification accuracy. We use Keras [16] library with Theano [9] backend for deep learning implementations. For Adadelta parameters, the default values of Keras 2 are used. The initializer for the convolutional layer is 'glorot uniform' [22] that draws samples from a truncated normal distribution. We use $L2$ regularization. The weight of the regularization for the layers in the left column of Table 4 is $w_{\text{reg}} = 2 \times (1e-4)$. Using Keras Model Check Point, the performance of the model on the validation set is checked at the end of each epoch and if there is an improvement over the best model so far, the weights are saved as the new best weights. At the end of all iterations, the best weights are loaded for the final model. In our experiments, the number of epochs is set to 200, and the batch size is set to 30. The accuracy of glitch classification for different durations is shown in Table 5.

We employ the "merged-view" model proposed in [8] and tune the hyper-parameters including number of layers and kernels for the current dataset. The accuracy obtained by the merged-view model is shown in the right most column of Table 5 and the detailed specification of the model is reported in the second column of Table 4 and drawn in Fig. 6. We set the regularization parameter to $w_{reg} = 2 \times (1e-4)$ for $L2$ regularization. All the other hyper-parameters and configuration settings for the "merged-view" model are the same as with the single view CNNs described earlier in this section.

3.4. Ultimate ensemble

In this section, we develop what we refer to an ultimate ensemble framework by combining a number of basic classifiers, presented in the previous sections, to solve the target glitch classification problem. We want to exploit the advantages of different classifiers in a unified model using ensembling techniques [34,44].

Initially, the following classifiers are selected as the basic classifiers:

- Merged-view one: merged-view model [8] of Section 3.3 with Model Check Point defined with respect to the accuracy over validation set.

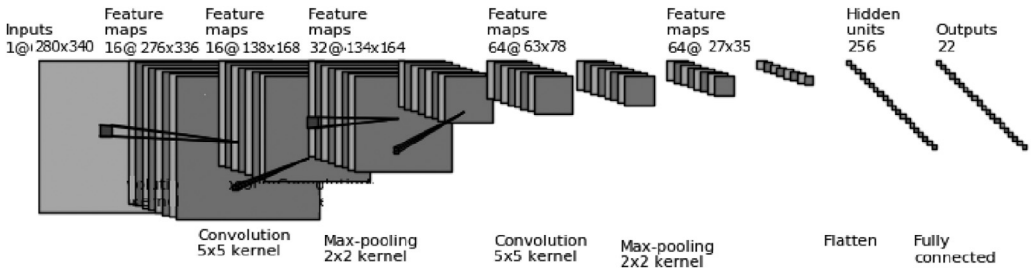
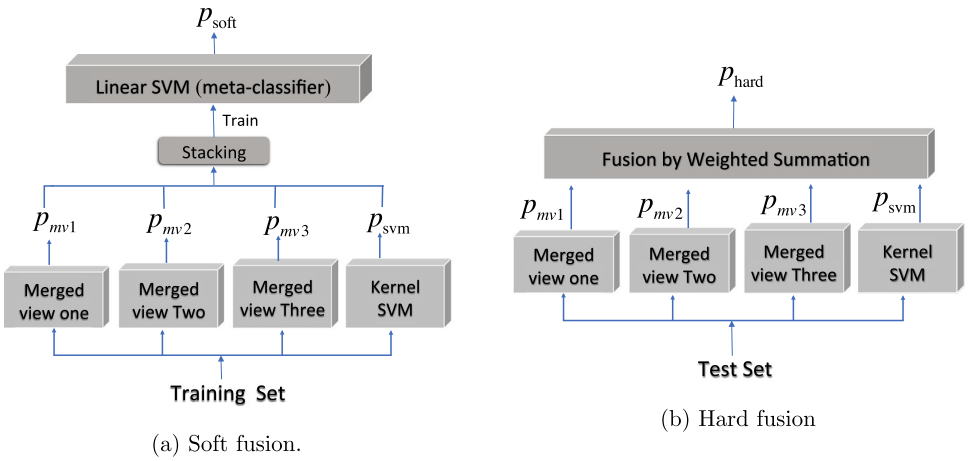


Fig. 6. The architecture of the merged-view model [8] tuned for this version of Gravity Spy dataset.

Table 6
Accuracy of ultimate ensembling models compared to the basic classifiers.

Merged-view one	Merged-view two	Merged-view three	Kernel SVM	Soft fusion	Hard fusion
97.67	97.44	97.51	97.12	98.06	**98.21**



(a) Soft fusion.

(b) Hard fusion

Fig. 7. The schematic representation of the ultimate ensemble of classifiers. In (a), a linear SVM is used as the meta-classifier and trained with the stacked probabilities vectors from basic classifiers. The trained linear SVM is then applied on test samples which are also in the stacked probability vector space to assign the predicted label to the test samples (not shown in this figure). In (b), there is no training step for hard fusion ensembling. Test samples are passed through the basic classifiers and the outputs are weighted sum to produce the final score for classification. The class corresponds to the maximum probability in the new probability vector is the predicted label for the test sample.

- Merged-view two: the same configuration as merged-view one but with Model Check Point defined with respect to the loss function value over the validation.
- Kernel SVM: trained for view with 1 second duration.

These models have shown the best performance among the presented baselines using traditional and deep neural network based approaches. Furthermore, to give the ensemble model more choices and provide additional diversity among the basic classifiers, we introduce another configuration of the merged view model and use it as the last basic classifier. We omit one of the convolutional layers with 64 kernels and its corresponding max-pooling layer (from the configuration given in the right column of Table 4) and also set the Model Check Point to track the accuracy over the validation set representing "merged-view three" in Table 6.

We further fuse the probabilistic outputs of the selected classifiers in two ways explained in the following sections.

3.4.1. Soft fusion

We perform score fusion by the "stacking with probability distributions" method [50]. A new classifier is trained over the probabilistic outputs of the basic classifiers. The new classifier should learn how to best combine the predictions of the primary models.

We use a linear SVM as the meta classifier for ensembling. The schematic representation of soft fusion method for ultimate ensembling is shown in Fig. 7(a). All training data samples are passed again through the selected basic classifiers and the probability vector over the 22 classes serve as the new feature vectors. All these feature vectors are concatenated to form the ultimate classifier input feature vector. The SVM is trained using the stacked probabilities feature vectors.

Feeding sample i to the soft fusion system, the outputs from the basic classifiers merged-view one, merged-view two, merged-view three, and kernel SVM are a 22×1 probability vectors p^i_{mv1}, p^i_{mv2} , p^i_{mv3}, and p^i_{svm}, respectively. For merged-view models, the vectors p^i_{mv1}, p^i_{mv2} , p^i_{mv3} are in fact feature vectors obtained at the output of a fully connected layer (see Eq. (8)). Stacking all probability vectors create an 88×1 feature vector $p^i_{stack} = [p^i_{mv1}, p^i_{mv2}, p^i_{mv3}, p^i_{svm}]^T$ which is used for training by the meta-classifier.

The performance of the soft fusion ensemble method compared to the basic classifiers (which are selected as the best individual classifier) is shown in Table 6. As can be seen the soft fusion method outperforms all basic classifiers. This is in line with our expectation about combining multiple classifiers to have the advantages of all of them in a unified model.

3.4.2. Hard fusion

With this method, we do not train a new classifier to combine the outputs of the basic classifiers; instead we strictly perform weighted summation over all the basic classifiers' outputs for test examples to obtain the hard fusion scores. The schematic representation of the hard fusion method is shown in Fig. 7(b). Hard fusion uses the following weighted sum to estimate the final score for test sample t

$$p^t_{hard} = \left[w_1 p^t_{mv1} + w_2 p^t_{mv2} + w_3 p^t_{mv3} + (1 - (w_1 + w_2 + w_3)) p^t_{svm} \right] \tag{10}$$

where p^t_{mv1} is a 22×1 vector obtained at the output of the fully connected layer of the merged-view one model as the test sample t is fed to the model. The same is true for all other merged-view models. The weights w_1, w_2, and w_3 are chosen in such a way so that each of them is between zero and one and $w_1 + w_2 + w_3 \leq 1$. The predicted class for test sample t is the one corresponding to the maximum value in the probability vector p^t_{hard}. The best values for w_1, w_2, and w_3 are obtained by discretizing each of them with a step 0.1 and considering all possible combinations while tracking the performance of the classifier over the validation set. The best result was obtained for $w_1 = w_2 = w_3 = 0.2$.

It is clear that there is no training here, since we do not use training data for the *ensembling part* of the system (they are used for training the basic classifiers though). Ultimate ensembling with hard fusion provides the best performance of 98.21% among all presented baselines. Although, we did expect that soft fusion is a more general form of the hard fusion approach, in practice hard fusion works better. The reason is probably due to the nature of the probability scores obtained at the output of the fully connected layer in deep neural networks which tend to have a pointy distribution which makes them less effective and generalizable for new feature vectors.

4. Conclusion

In this paper, we present the Gravity Spy dataset for public release, mainly for the machine learning community. This dataset includes 8583 of images of LIGO glitches and the specifications for 22 glitch classes. We further built and presented the standard split of data for training, validation and testing sets to make the comparison between various glitch classification methods feasible. Employing state-of-the-art classification methods, we developed several baselines for the supervised glitch classification problem. For the best classifier, which is an ensemble of deep neural network models and an SVM model, we obtained 98.21% accuracy on the standard test set. All classifiers were trained using the standard training set and the validation set was used for tuning the hyper-parameters wherever needed (the validation set samples were never added to the training set). We expect that the material presented in this paper will be beneficial to future studies. For example, the described herein Gravity Spy dataset along with the classification algorithms can be utilized to compare any new classification algorithms and accurately access their performance. The glitch classes present in LIGO data evolve with time and we expect to produce and study more such datasets in the future. Even in this case one will be able to fall back to the results presented in this paper and evaluate appropriately any future developments.

Acknowledgments

We would like to acknowledge all the members of the Gravity Spy project for all their efforts and time on this project. In addition, the team would like to thank the Detector Characterization working group of the LSC for useful comments and suggestions. We would like to thank our undergraduate research team who helped in the building and creation of the initial training set of glitches: Luke Calian, Jessie Duncan, Ethan Marx, Isa Patane, Leah Perri, and Ben Sandeen. Gravity Spy is partly supported by the National Science Foundation, award INSPIRE 15–47880.

References

[1] J. Aasi, J. Abadie, B.P. Abbott, R. Abbott, T. Abbott, M.R. Abernathy, T. Accadia, F. Acernese, C. Adams, T. Adams, et al., Characterization of the LIGO detectors during their sixth science run, Class. Quant. Grav. 32 (11) (2015) 115012, doi:10.1088/0264-9381/32/11/115012.

[2] B.P. Abbott, R. Abbott, T.D. Abbott, M.R. Abernathy, F. Acernese, K. Ackley, M. Adamo, C. Adams, T. Adams, P. Addesso, et al., Characterization of transient noise in advanced LIGO relevant to gravitational wave signal GW150914, Class. Quant. Grav. 33 (13) (2016) 134001, doi:10.1088/0264-9381/33/13/134001.

[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, et al., GW151226: Observation of gravitational waves from a 22-Solar-Mass binary black hole coalescence, Phys. Rev. Lett. 116 (24) (2016) 241103, doi:10.1103/PhysRevLett.116.241103.

[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., Observation of gravitational waves from a binary black hole merger, Phys. Rev. Lett. 116 (6) (2016) 061102, doi:10.1103/PhysRevLett.116.061102.

[5] B.P. Abbott, R. Abbott, T.D. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R.X. Adhikari, V.B. Adya, et al., GW170104: Observation of a 50-Solar-Mass binary black hole coalescence at redshift 0.2, Phys. Rev. Lett. 118 (22) (2017) 221101, doi:10.1103/PhysRevLett.118.221101.

[6] T. Accadia, F. Acernese, F. Antonucci, P. Astone, G. Ballardin, F. Barone, M. Barsuglia, T.S. Bauer, M. Beker, A. Belletoile, et al., Noise from scattered light in virgo's second science run data, Class. Quant. Grav. 27 (19) (2010) 194011.

[7] S.-I. Amari, S. Wu, Improving support vector machine classifiers by modifying kernel functions, Neural Netw. 12 (6) (1999) 783–789.

[8] S. Bahaadini, N. Rohani, S. Coughlin, M. Zevin, K. Vicky, A. Katsaggelos, Joint deep multi-view models for glitch classification, in: Proceedings of the Forty-second IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

[9] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio, Theano: a CPU and GPU math expression compiler, in: Proceedings of the Python for Scientific Computing Conference (SciPy), 2010. Oral Presentation

[10] 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, X. Wang, Application of machine learning algorithms to the study of noise artifacts in gravitational-wave data, Phys. Rev. D 88 (2013) 062003, doi:10.1103/PhysRevD.88.062003.

[11] C. Biwer, D. Barker, J.C. Batch,, J. Betzwieser, R.P. Fisher, E. Goetz, S. Kandhasamy, S. Karki, J.S. Kissel, A.P. Lundgren, and others, Validating gravitational wave detections: The Advanced LIGO hardware injection system, Phys. Rev. D 95 (6) (2017) 062002.

[12] K.D. Borne, L. Fortson, P. Gay, C. Lintott, M.J. Raddick, J. Wallin, The zooniverse, in: Proceedings of the AGU Fall Meeting Abstracts, 2009.

[13] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: experiences from the scikit-learn project, in: Proceedings of the ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.

[14] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov. 2 (2) (1998) 121–167.

[15] S. Chatterji, L. Blackburn, G. Martin, E. Katsavounidis, Multiresolution techniques for the detection of gravitational-wave bursts, Class. Quant. Grav. 21 (20) (2004) S1809. http://stacks.iop.org/0264-9381/21/i=20/a=024.

[16] F. Chollet, keras, 2015. https://github.com/fchollet/keras.

[17] T. Dharani, I.L. Aroquiaraj, A survey on content based image retrieval, in: Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), IEEE, 2013, pp. 485–490.

[18] S. Dwyer, Change in OMC length gain helps with 1083 Hz glitches, 2017. Advanced LIGO electonic log 33104. https://alog.ligo-wa.caltech.edu/aLOG/index.php?callRep=33104.

[19] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, LIBLINEAR: a library for large linear classification, J. Mach. Learn. Res. 9 (2008) 1871–1874.

[20] B. Gateley, End station instrument air compressors, 2015. Advanced LIGO electonic log 22081. https://alog.ligo-wa.caltech.edu/aLOG/index.php?callRep=22081.

[21] D. George, H. Shen, E.A. Huerta, Deep transfer learning: a new deep learning glitch classification method for advanced LIGO, ArXiv e-prints: arXiv:1706.07446v1 (2017).

[22] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[23] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the International Conference on Learning Representations (ICLR), 2015.

[24] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[25] C. Kwak, A. Clayton-Matthews, Multinomial logistic regression, Nurs. Res. 51 (6) (2002) 404–410.

[26] S. Lawrence, C.L. Giles, A.C. Tsoi, A.D. Back, Face recognition: a convolutional neural-network approach, IEEE Trans. Neural Netw. 8 (1) (1997) 98–113.

[27] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[28] J. Aasi, B.P. Abbott, R. Abbott, T. Abbott, M.R. Abernathy, K. Ackley, C. Adams, T. Adams, P. Addesso, et al., The LIGO Scientific Collaboration, Advanced LIGO, Class. Quant. Grav. 32 (7) (2015) 074001, doi:10.1088/0264-9381/32/7/074001.

[29] A. Lundgren, New glitch class: paired doves, 2016. Advanced LIGO electonic log 27138. https://alog.ligo-wa.caltech.edu/aLOG/index.php?callRep=27138.

[30] R. Lynch, S. Vitale, R. Essick, E. Katsavounidis, F. Robinet, An information-theoretic approach to the gravitational-wave burst detection problem, ArXiv e-prints: 1511.05955 (2015).

[31] L.V.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.

[32] F. Mansoori, E. Wei, Superlinearly convergent asynchronous distributed network newton method, in: Proceedings of the 2017 IEEE Fifty-sixth Annual Conference on Decision and Control (CDC), 2017, pp. 2874–2879.

[33] K. Mertsalov, M. McCreary, Document classification with support vector machines, 2009.

[34] R.S. Michalski, J.G. Carbonell, T.M. Mitchell, Machine Learning: An Artificial Intelligence Approach, Springer Science & Business Media, 2013.

[35] S. Mukherjee, R. Obaid, B. Matkarimov, Classification of glitch waveforms in gravitational wave detector characterization, J. Phys. Conf. Ser. 243 (1) (2010) 012006, doi:10.1088/1742-6596/243/1/012006.

[36] N. Mukund Menon, S. Abraham, S. Kandhasamy, S. Mitra, N. Philip, Transient classification in LIGO data using difference boosting neural networks, Phys. Rev. D 95 (2016).

[37] V. Noroozi, A. Hashemi, M. Meybodi, Alpinist cellularde: a cellular based optimization algorithm for dynamic environments, in: Proceedings of the Fourteenth Annual Conference Companion on Genetic and Evolutionary Computation, ACM, 2012, pp. 1519–1520.

[38] V. Noroozi, L. Zheng, S. Bahaadini, S. Xie, P.S. Yu, Seven: deep semi-supervised verification networks, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, AAAI Press, 2017, pp. 2571–2577.

[39] 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, C.C. Wipf, Improving the data quality of advanced LIGO based on early engineering run results, Class. Quant. Grav. 32 (24) (2015) 245005, doi:10.1088/0264-9381/32/24/245005.

[40] J. Powell, D. Trifirò, E. Cuoco, I.S. Heng, M. Cavaglià, Classification methods for noise transients in advanced gravitational-wave detectors, Class. Quantum Grav. 32 (21) (2015) 215012.

[41] J. Powell, D. Trifirò, E. Cuoco, I.S. Heng, M. Cavaglià, Classification methods for noise transients in advanced gravitational-wave detectors, Class. Quant. Grav. 32 (21) (2015) 215012, doi:10.1088/0264-9381/32/21/215012.

[42] S. Rampone, V. Pierro, L. Troiano, I.M. Pinto, Neural network aided glitch-Burst discrimination and glitch classification, Int. J. Modern Phys. C 24 (2013) 1350084, doi:10.1142/S0129183113500848.

[43] F. Robinet, Omicron: an algorithm to detect and characterize transient events in gravitational-wave detectors, 2014. VIRGO Technical Document VIR-0545A-14. https://tds.virgo-gw.eu/.

[44] L. Rokach, Ensemble-based classifiers, Artif. Intell. Rev. 33 (1) (2010) 1–39.

[45] R. Schofield, Damping reduced in-air velocity of swiss cheese baffle by more than an order of magnitude, 2017. Advanced LIGO electonic log 36147. https://alog.ligo-wa.caltech.edu/aLOG/index.php?callRep=36147.

[46] A. Sharifi, V. Noroozi, M. Bashiri, A. Hashemi, M. Meybodi, Two phased cellular PSO: a new collaborative cellular algorithm for optimization in dynamic environments, in: Proceedings of the 2012 IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–8.

[47] J. Smith, The 50Hz glitches in DARM: EX mains glitches coupling into EX seismic, 2015. Advanced LIGO electonic log 21436. https://alog.ligo-wa.caltech.edu/aLOG/index.php?callRep=21436.

[48] J.R. Smith, T. Abbott, E. Hirose, N. Leroy, D. MacLeod, J. McIver, P. Saulson, P. Shawhan, A hierarchical method for vetoing noise transients in gravitational-wave detectors, Class. Quant. Grav. 28 (23) (2011) 235005. http://stacks.iop.org/0264-9381/28/i=23/a=235005.

[49] 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 others, Binary black hole mergers in the first advanced LIGO observing run, Phys. Rev. X 6 (4) (2016) 041015.
[50] K.M. Ting, I.H. Witten, Issues in stacked generalization, J. Artif. Intell. Res. 10 (1999) 271–289.
[51] J. Watt, R. Borhani, A. Katsaggelos, Machine Learning Refined: Foundations, Algorithms, and Applications, Cambridge University Press, 2016.
[52] M.D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701 (2012).
[53] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, M. Cabero, K. Crowston, A. Katsaggelos, S. Larson, et al., Gravity spy: integrating advanced ligo detector characterization, machine learning, and citizen science, Class. Quant. Grav. 34 (6) (2017) 064003.
[54] L. Zheng, B. Cao, V. Noroozi, P.S. Yu, N. Ma, Hierarchical collaborative embedding for context-aware recommendations, in: Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 867–876.