Classical and Quantum Gravity

PAPER

Image-based deep learning for classification of noise transients in gravitational wave detectors

To cite this article: Massimiliano Razzano and Elena Cuoco 2018 *Class. Quantum Grav.* **35** 095016

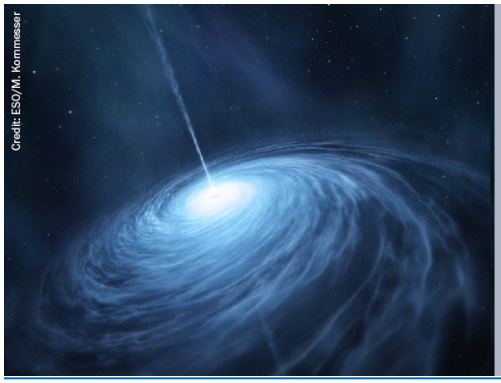
View the article online for updates and enhancements.

Related content

- Classification methods for noise transients in advanced gravitational-wave detectors
  Jade Powell, Daniele Trifirò, Elena Cuoco et al.
- A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update
  F Lotte, L Bougrain, A Cichocki et al.
- Classification methods for noise transients in advanced gravitational-wave detectors II: performance tests on Advanced LIGO data
  Jade Powell, Alejandro Torres-Forné, Ryan Lynch et al.

Recent citations

- Enhancing gravitational-wave science with machine learning
  Elena Cuoco *et al*
- Core-Collapse supernova gravitational-wave search and deep learning classification
  Alberto Iess *et al*



AMERICAN ASTRONOMICAL SOCIETY

IOP | ebooks™

Your first choice for astronomy, astrophysics, solar physics, and planetary science ebooks.

Start exploring the collection–download the first chapter of every title for free.

This content was downloaded from IP address 136.233.197.195 on 28/12/2020 at 04:09

IOP Publishing Classical and Quantum Gravity

Class. Quantum Grav. **35** (2018) 095016 (21pp) https://doi.org/10.1088/1361-6382/aab793

Image-based deep learning for classification of noise transients in gravitational wave detectors

Massimiliano Razzano[1,3] **and Elena Cuoco**[2,3]

[1] Department of Physics, University of Pisa, Pisa I-56127, Italy
[2] European Gravitational Observatory (EGO), I-56021 Cascina, Pisa, Italy
[3] Istituto Nazionale di Fisica Nucleare, Sezione di Pisa, Pisa I-56127, Italy

E-mail: massimiliano.razzano@unipi.it

Received 31 October 2017, revised 9 March 2018
Accepted for publication 19 March 2018
Published 5 April 2018

Abstract

The detection of gravitational waves has inaugurated the era of gravitational astronomy and opened new avenues for the multimessenger study of cosmic sources. Thanks to their sensitivity, the Advanced LIGO and Advanced Virgo interferometers will probe a much larger volume of space and expand the capability of discovering new gravitational wave emitters. The characterization of these detectors is a primary task in order to recognize the main sources of noise and optimize the sensitivity of interferometers. Glitches are transient noise events that can impact the data quality of the interferometers and their classification is an important task for detector characterization. Deep learning techniques are a promising tool for the recognition and classification of glitches. We present a classification pipeline that exploits convolutional neural networks to classify glitches starting from their time-frequency evolution represented as images. We evaluated the classification accuracy on simulated glitches, showing that the proposed algorithm can automatically classify glitches on very fast timescales and with high accuracy, thus providing a promising tool for online detector characterization.

Keywords: gravitational waves, interferometers, deep learning

(Some figures may appear in colour only in the online journal)

1. Introduction

The detection of gravitational waves by the Advanced Laser Interferometer Gravitational Observatory (LIGO) has started the era of gravitational wave astronomy and opened a new window on the Universe. Among the events detected so far, most of them are consistent with

the coalescence of binary black holes (e.g. GW150914, GW151226, GW1701014, GW170814 [1–4]), while GW170817 [5] is interpreted as the signal from coalescing neutron stars.

The coalescence of compact binary systems is a primary source of gravitational waves. In particular, systems containing at least one neutron star are believed to be powerful electromagnetic emitters, thus being prime targets for multimessenger investigations. Core-collapse supernovae are also expected to emit bursts of gravitational waves [6]. Furthermore, continuous, periodic gravitational wave emission is expected from asymmetric neutron stars [7], as well as a continuous gravitational-wave stochastic background.

Advanced LIGO [8] and Advanced Virgo [9] are second-generation laser interferometers aimed at reaching a sensitivity $\sim$10 times better that of the previous LIGO and Virgo detectors. This increase in sensitivity will offer the possibility to explore a larger volume of space and increase the rate of detected signals. In order to reach the target sensitivity, both LIGO and Virgo detectors underwent significant upgrades on all major subsystems, including optics, suspensions, and seismic attenuation systems.

Advanced LIGO began operations in 2015 and has completed two observing runs so far, called O1 (12 September 2015–19 January 2016) and O2 (30 November 2016–25 August 2017). The construction of Advanced Virgo has been completed at the end of 2016 and the interferometer has joined O2 on 1 August 2017 [10].

The detection of gravitational wave transients is based on two different strategies implemented by the LIGO and Virgo Collaborations with dedicated pipelines. When the gravitational waveform is known with a good accuracy, as for a compact binary coalescence, the optimal detection is achieved with matched filtering with template banks of expected waveforms (e.g. [11, 12]). When the shape of gravitational waveform is unknown, excess-power strategies has been developed, such as coherent network algorithms [13].

All search methods are limited by the detector noise, which usually is non stationary and frequency-dependent. At low frequencies ($<$10 Hz) the main contribution to the noise is due to seismic ground motion and to the gradient of the local gravitational field. At higher frequencies the thermal noise of the subsystems, e.g. the suspensions of the mirrors, plays a dominant role. Above $\sim$200 Hz the most important limitation to the detector sensitivity is the shot noise of the laser light circulating in the detector cavities [14].

The noise in the interferometers has also transient components produced by interactions among detector subsystems or with the surrounding environment. These transient noise signals, called 'glitches' have an impact on the data quality of the interferometers and in some cases could mimic the time-frequency behavior of astrophysical signals. However, real gravitational wave transient events produce consistent waveforms that are delayed due to the propagation between detectors, and so they can be discriminated from noise by comparing data in independent detectors [15, 16]. Near-simultaneous glitches can still mimic the signature of a gravitational wave signals, therefore it is also important to look for noise transients in auxiliary channels of each detector, that are not sensitive to gravitational waves but can provide useful information on the status of the environment and of the interferometer.

The identification and characterization of these glitches is important in order to investigate the impact of transient noise on the detectors. It is very useful to produce a classification of glitches according to their time and frequency evolution, in order to group them in families of similar morphology.

Once a glitch family has been identified, it is possible to perform further investigations to establish its origin and prepare custom data quality flags in order to reduce their impact on the detector performance [15].

Manual inspection of the glitch parameters is time consuming and can be subject to errors, therefore is not the optimal strategy in the era of advanced interferometers, since the increased

complexity and sensitivity of the detectors will lead to large quantities of glitches of various morphologies.

Different methods have been developed to automatically classify glitches in advanced gravitational wave detectors. For instance, in [17], three algorithms were tested on Advanced LIGO engineering run data: (1) PCAT, based on principal component analysis [18], (2) LALInferenceBurst, based on Bayesian parameter estimation [19], and (3) WDF-ML, that couples a wavelet detection filter [20] with an unsupervised machine learning clustering based on Gaussian mixture model algorithm [21]. In [22], a supervised Bayesian classifier called difference boosting neural network (DBNN) has been tested on LIGO S6 data, also showing very good performance.

Among the possible approaches to the automatic analysis and classification of glitches, machine learning looks very promising. Machine learning is a field of computer science based on algorithms that can learn from data and make predictions on it [23]. This approach is extensively used in many applications of big data analysis, including marketing, speech recognition and computer vision, and is becoming very popular in many field of scientific research, including the study of electron structure [24, 25], quantum many-body systems [26], molecular models [27, 28], and particle physics [29].

Machine learning algorithms can be divided in two main categories: supervised ones, which perform classification or regression tasks by learning from samples of labeled data, and unsupervised ones, which learn relationships automatically from non-labeled data.

Deep learning is an area of machine learning that combines the architecture of the artificial neural networks with the power of the machine learning algorithms (see, e.g. [30]). By stacking many layers of artificial neurons in a neural network architecture, it is possible to build very efficient tools that perform classification or regression tasks.

Deep learning algorithms have been suggested for applications in gravitational wave physics [31], highlighting the potential of convolutional neural networks (CNNs). In particular, the time-frequency behavior of glitch signals can be converted to images and given as input to CNNs, which show very good performance in image analysis and recognition tasks [32].

Various baseline methods exist to classify images, such as minimum distance, maximum likelihood, support vector machines, random forest [31], or fully connected neural networks. However, we decided to explore the potential of deep learning because of the complexity of gravitational wave data, as done in previous works (e.g. [17, 31]). In particular, transient noise glitches often originate from complex nonlinear couplings between the channels of the detector, thus resulting in very complex time-frequency morphologies. This is supported by the analysis of real data, where glitches appear to come in many different classes. Furthermore, the size of the time-frequency images that we are using to classify glitches is too large for a simpler method (e.g. fully connected networks), given the constraints on our hardware. The exquisite capability of deep networks to extract features from large images with minimal setup and parameter tuning, and their power in multiclass problem, makes them an optimal algorithm for our purpose. Training a deep network could take time, but this is not an issue for our analysis. In fact, to classify a set of glitches without any requirement on speed, the training can be performed once. In case the classification algorithm is running in low-latency mode on new coming data, and thus speed is fundamental, the training can be performed periodically on a dedicated machine, and the resulting model can be uploaded to a second machine that is dedicated only to the classification.

In this paper we introduce a new tool for the classification of transient noise in gravitational wave interferometers, that exploits the power of image processing with CNNs. The structure of the paper is the following: in section 2 we introduce the deep learning algorithms, focusing on the CNNs and their utility in astrophysics. In section 3 we describe the proposed analysis

pipeline and algorithms, while section 4 is focused on the simulated transients prepared to test the method. In section 5 we report the summary of the performance, that are further discussed in section 6. In section 7 we highlight the possible extensions and prospects of the suggested method.

2. Deep learning for image recognition

Deep learning has found many applications in the analysis of big data, in particular on complex tasks such as computer vision [33], image processing [34], speech recognition [35] and natural language processing [36]. Furthermore, deep learning algorithms for image analysis and recognition have been successfully tested and implemented in many fields of astrophysics, including galaxy classification [37], cosmology [38], and asteroseismology [39].

The base element of a neural network is the artificial neuron, introduced in 1943 by McCulloch and Pitts as a result of their efforts to understand and design artificial networks [40]. A later work introduced the *perceptron* rule [41], an algorithm that automatically learns the optimal weights coefficients w_i that are multiplied by the components of an input vector $\vec{\mathbf{x}}$ in order to decide whether a neuron will fire or not.

In fact, each perceptron computes the *activation function* $\phi(z)$, where z is the so-called *net input* defined as:

$$z = \vec{\mathbf{w}} \cdot \vec{\mathbf{x}} + b = \sum_{i=1}^{n} w_i x_i + b. \tag{1}$$

In the context of a single-neuron network, $\vec{\mathbf{x}}$ represents the vector containing the properties, or *features*, of a data sample. For image analysis, the features could be the intensity values of each pixel. In more complex networks, the vector $\vec{\mathbf{x}}$ is the outputs of the other perceptrons. The coefficient b is the *bias* term and is introduced to shift the decision boundary, which can be useful to improve the accuracy of the algorithm. In the basic implementation of the perceptron, the activation function is the Heaviside step function $\theta(z)$ and can be used to implement a classifier among two classes corresponding to the two output values of that function.

The interest in neural networks was boosted by the development of the *backpropagation* technique for efficient training of neural networks [42]. The introduction of the *multilayer perceptron*, an artificial network made of stacked layers of neurons, overcame the limitations of the perceptron. These architectures are also called *feedforward neural networks* because they process the input forward to the output without forming any internal cycle.

Furthermore, the universal approximation theorem [43] states that a 3-layer feedforward network with one hidden layer and finite number of neurons can approximate any continuous function. Therefore, deep neural networks can be used to model and approximate complex functions, whose output could be discrete or continuous, for classification or regression tasks respectively.

Convolutional neural networks (CNNs) are a subset of feedforward deep neural networks developed to improve the performance of deep learning on image processing [44]. The key element in CNNs are *convolutional* layers, where neurons are optimized to capture specific details in an image. In order to achieve this, the neurons are sharing weights within a kernel filter. During the training, the weights are computed by sliding the filter across the neurons in the layer, resulting in a kernel convolution. The size of the kernel is related to the scale of the details to be looked for in the image. Sharing the weights greatly reduces the number of neurons required to analyze an image, with a net gain in computational resources with respect

to the standard, fully-connected, deep networks. Furthermore, sharing weight also introduces a regularization, thus preventing the algorithm from potential overfitting.

Each convolutional layer is coupled with a *pooling* layer that acts as a subsampling operator by reducing the size of the output of a convolutional layer.

In order to reduce the problem of overfitting, CNNs also feature *dropout* layers, that randomly ignore a set of neurons during training [45]. The dropout is a regularization strategy widely used and has the overall effect of making the CNN less sensitive to the specific weights of neurons.

Once a CNN has been trained on a set of images, it is capable of performing classification or other tasks in very short time, and is therefore very well suited for tasks that require low-latency processing. Thanks to their flexibility and capability of processing large images, CNNs have been widely implemented in various fields of medical physics [46], and it is now being tested in projects involving image analysis in astronomy [47]. The accuracy and speed of CNNs are also very important for tasks requiring low latency, and they have been proposed also for the automatic search for transient phenomena in gravitational wave data [31, 48].

3. Description of the pipeline

We implemented a CNN-based classification pipeline for the automatic classification of glitches and other noise transients in advanced gravitational wave detectors. The input of the pipeline are images representing the time-frequency evolution of the glitches For instance, this can be derived from the signal in $h(t)$, the channel representing the output of the interferometers after proper vetoes and calibrations are applied. For a transient signal of astrophysical origin, $h(t)$ measures the gravitational wave strain induced by the passage of gravitational waves, and can be contaminated by glitches.

3.1. Whitening procedure

Before applying any detection and classification pipeline, the data should be conditioned. The usual procedure is the whitening one [49, 50], which can remove the contribution of Gaussian colored detector noise. Many pipelines use whitening in frequency domain, while the whitening here applied is based on time-domain procedure, using an autoregressive (AR) fit to the data.

An auto-regressive process $x[n]$ of order P with parameter a_k, from here after AR(P), is characterized by the relation

$$x[n] = \sum_{k=1}^{P} a_k x[n-k] + \sigma w[n]\,, \tag{2}$$

being $w[n]$ a white normal process.

The problem of determining the AR parameters is the same of that of finding the optimal 'weights vector' $\mathbf{w} = w_k$, for $k = 1, ...P$ for the problem of linear prediction [51]. In the linear prediction we would predict the sample $x[n]$ using the P previous observed data $\mathbf{x}[n] = \{x[n-1], x[n-2] \ldots x[n-P]\}$ building the estimate as a transversal filter:

$$\hat{x}[n] = \sum_{k=1}^{P} w_k x[n-k]\,. \tag{3}$$

We choose the coefficients of the linear predictor by minimizing a cost function that is the mean squares error $\epsilon = \mathcal{E}[e[n]^2]$ ($\mathcal{E}$ is the operator of average on the ensemble), being

$$e[n] = x[n] - \hat{x}[n] \tag{4}$$

the error we make in this prediction and obtaining the so called Normal or Wiener–Hopf equations

$$\epsilon_{\min} = r_{xx}[0] - \sum_{k=1}^{P} w_k r_{xx}[-k]\,, \tag{5}$$

which are identical to the Yule–Walker equations [51] used to estimated the parameters a_k from autocorrelation function with $w_k = -a_k$ and $\epsilon_{\min} = \sigma^2$

This relationship between AR model and linear prediction assures us to obtain a filter which is stable and causal [51], so we can use the AR model to reproduce stable processes in time-domain. It is this relation between AR process and linear predictor that becomes important in the building of whitening filter. The tight relation between the AR filter and the whitening filter is clear in the figure 1. The figure describes how an AR process colors a white process at the input of the filter if you look at the picture from left to right. If you read the picture from right to left you see a colored process at the input that pass through the AR inverse filter coming out as a white process. When we find the P parameters that fit a PSD of a noise process, what we are doing is to find the optimal vector of weights that let us reproduce the process at the time n knowing the process at the P previous time. All the methods that involve this estimation try to make the error signal (see equation (4)) a white process in such a way to throw out all the correlation between the data (which we use for the estimation of the parameters). Suppose to have a sequence $x[n]$ of data characterized by an autocorrelation $r_{xx}[n]$ which is not a delta function, and that you need to remove all the correlation, making it a white process. The idea is to model $x[n]$ as an AR process, find the AR parameters and use them to whiten the process.

Using a lattice structure [49] we can implement the whitening filter in time domain. This is a procedure which is used for other pipeline [17, 52] and which can be implemented also in adaptive way [53] taking care of non stationary noise.

3.2. Image preparation

If the images of the glitches are not available, we introduce an image preparation stage, that converts the whitened time series to images showing the time-frequency evolution of the glitches. In order to test the pipeline, in this paper we have built images using simulated time series that cover 2 s around each glitch. A complementary, multi-view architecture combining different timescales has also been proposed [54]. The images are built from the Fourier spectrograms of the time series. Other algorithms can be used to produce the images, such as the constant Q transform or Wavelet transforms. The spectrograms are then converted to 32-bit gray scale images, by rescaling the spectrogram values in the range 0–2^{32}.

The images can be further processed, if needed, in order to reduce the residual noise features and enhance the details. For instance, in order to further enhance the contrast in the images we can apply, where needed, a normalization in the image histogram that ignores the pixel values outside the 2% and the 98% percentiles.

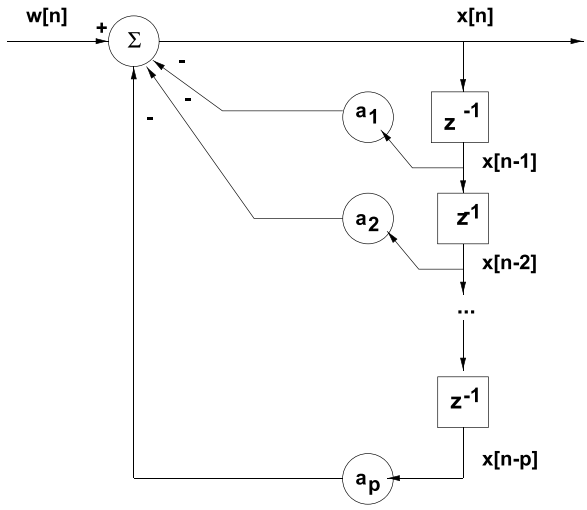


Figure 1. Whitening filter and AR filter.

3.3. Detection and classification

The main goal of this algorithm is the classification, however CNNs can be also used to detect glitches in the data stream. The detection of a glitch signal can be seen as an *anomaly detection* problem, that can be tackled with supervised or unsupervised algorithms. In our framework of supervised CNNs, a simple way to implement the anomaly detection is using a 'signal versus noise' binary classification approach[4]. To this purpose, the images containing glitches can be grouped in a SIGNAL class, and a set of time-frequency images of data stream with no glitches can be used to produce a set of images for a NOISE class, since they contain only the non transient component of the detector noise. This binary classification problems work well for standard machine learning algorithms, such as support vector machines or random forest.

Since CNNs work smoothly on predicting many classes, we implemented in our pipeline a multi-class classification. Given a number of glitch classes N_c, we have produced a set of glitch-free images to populate a NOISE class, so that the pipeline can perform a $N_c + 1$ classification of glitches or noise. For each glitch image, the pipeline computes the probability of belonging to one of the N_c glitch class or to the NOISE class[5]. For the training phase we split our glitch set in training, validation, and test set with the ratio 70:15:15.

In a realistic situation, it can happen that the number of images is different for each class [54]. To fix this class imbalance problem, we implemented a balancing step based on image augmentation procedures [55]. Given the appearance of the time-frequency images of glitches, we perform the augmentation by simply shifting the images by a small amount, typically of $\sim$10% of the size of the image. We have found that rotating and flipping the images is not very important on glitch images, that show always the same orientation and direction in time.

Once the training phase has been completed, the model and the weights computed by the CNN are saved on disk. Using this model, the pipeline can perform classification starting from the time series or the image of the new glitches. The output of this step is the class of the glitch, with the associated probability of belonging to that class.

[4] From now on, by *signal* we mean a transient glitch signal, and for *noise* we mean the stationary, non-transient, detector noise

[5] Even if the presence of a glitch in the data stream has been already assessed by another external detection algorithm, the presence of a NOISE class is useful to cross check the robustness of the detection.

3.4. Implementation of the architecture

The pipeline has been developed using the Python scripting language, and we used high-performance numerical libraries such as NumPy and SciPy [56, 57]. The CNN architecture has been built using the Python-based libraries Theano [58] and Keras [59], which provide high performance and easy implementation of prototypes. The code has been developed and tested on a GPU NVIDIA GeForce GTX 780 running CUDA 8.0 and and the accelerated libraries cuDNN developed for deep learning applications [60].

We have tested difference CNN configurations and we selected an architecture based on four main blocks. The first block is made of a 16 and a 32-deep layer, while the second and the third block contains 2 layers with depth of 64 and 128 respectively. Each convolutional kernel has a 3×3 and a *ReLU* activation function. Each block contains Maximum Pool layers (Keras MaxPooling2D) and is followed by a dropout layer (Keras Dropout). The last block is made of a fully connected layer (Keras Dense) with 512 neurons and by an output layer with a size equal to the number of classes N_c and with a *SoftMax* activation function. If the NOISE class is added, the output size is increased by one unit. Figure 2 shows the blocks in the CNN used for the pipeline.

In order to show the classification capabilities of this pipeline, it is important to compare its performance with that of simpler architectures and use a common dataset for the comparison. The datasets used in previous works (e.g. those cited in section 1) are in some cases too small for a deep learning algorithms or not available. Therefore, in order to assess the performance of our algorithm in this paper we have performed a comparison using the dataset of simulated glitches described in next section. The detail and results of this comparison will be presented in section 5

4. Glitch simulations

In order to test the classification algorithm, we prepared simulated a set of time series that include glitches with different shapes and intensities. This approach, also adopted in previous works [17, 22], allows to evaluate classification performance depending on the parameters of the simulated glitches.

The simulated glitches have been added to a Gaussian noise generated from the sensitivity curve of real interferometers. Here we used the sensitivity of LIGO Hanford detector (H1) available online[6].

4.1. Glitch families

We have simulated six families of signals, that approximate the time-frequency evolution of main noise transients observed in the real interferometers [15].

Each glitch is described by a time series $h(t)$. We also included a family of chirp-like transients, in order to show the potential of the CNNs in distinguishing transients of astrophysical origin and of known waveform, such as those produced by coalescing binaries. Therefore, we modeled these transients using a $h_+(t)$ and $h_\times(t)$ component, that are combined with the detector response to get the input signal $h(t) = F_+(\theta, \phi)h_+(t) + F_\times(\theta, \phi)\, h_\times(t)$, where $F_+(\theta, \phi)$ and $F_\times(\theta, \phi)$ are the antenna factors of the interferometer. In this case, θ and ϕ represent a random direction in the sky. Since chirp-like glitches have been observed in real data

[6] Data publicly available at the LIGO Open Science Center https://losc.ligo.org/s/events/GW150914/P150914/fig1-observed-H.txt

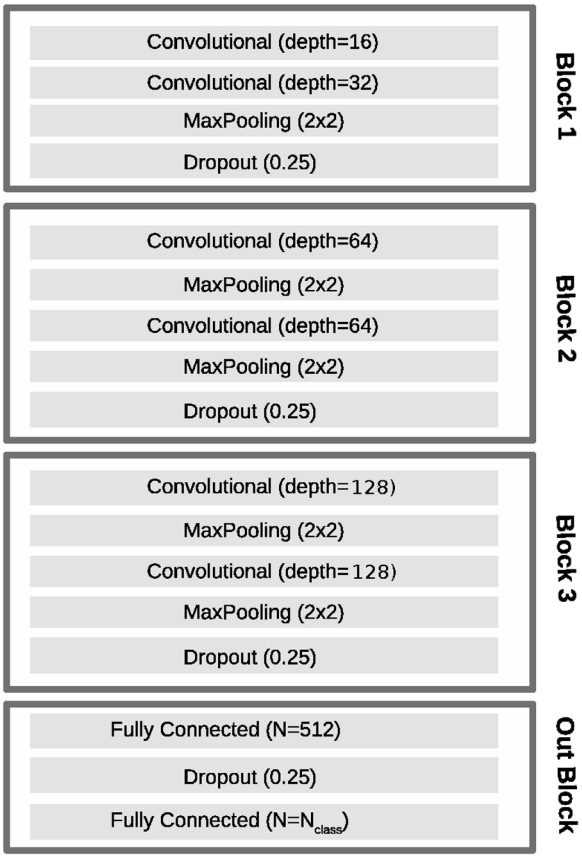


Figure 2. Diagram showing the structure of the CNN used in this work (For more detail see section 3.4). The size of the convolutional kernels is always 3×3. Here the output size is the number of classes N_c, but becomes $N_c + 1$ if the NOISE class is used (see text for more details).

[15], these simulations also help to show the capability of recognizing this glitch morphology. We then scaled the amplitude h_0 of each simulated glitch by its root sum squared amplitude $h_{\rm rss} = \int_{-\infty}^{\infty}(|h_+(t)|^2 + |h_\times(t)|^2)\mathrm{d}t$ [61].

The glitch families are described below in more detail, while figure 3 shows a sample gallery of glitches belonging to these families, that we describe here:

4.1.1. Gaussian (GAUSS). These glitches are described by a simple Gaussian waveform and model the short spikes observed in previous works of detector characterization [15]. They have a characteristic width τ, a peak time t_0, and can be modeled as:

$$h(t) = h_0 \exp\left(-\frac{(t-t_0)^2}{2\tau^2}\right). \tag{6}$$

4.1.2. Sine-Gaussian (SG). These glitches are described by a sinusoid whose amplitude is modulated by a Gaussian. They are centered at a frequency f_0 and their width τ is related to f_0 and to the quality factor Q as $\tau = \frac{Q}{\sqrt{2}\pi f_0}$.

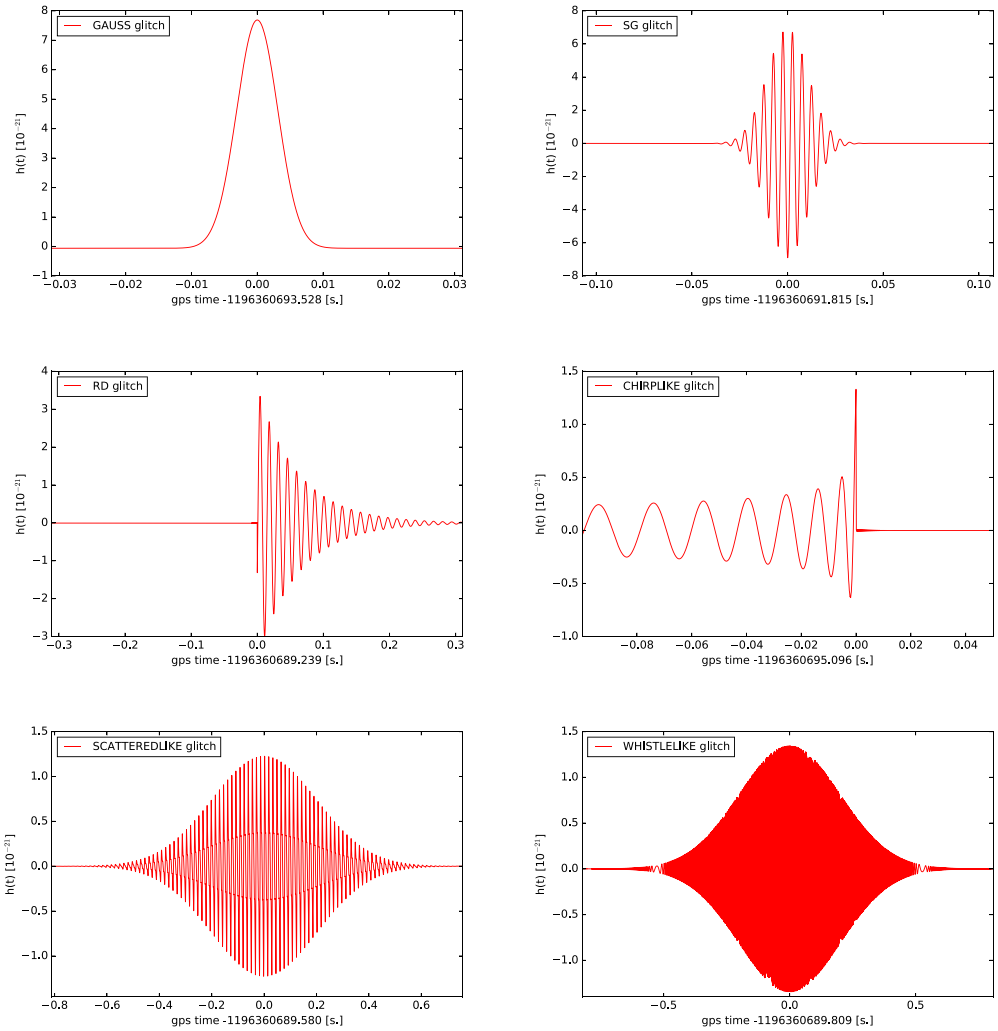


Figure 3. Gallery of simulated glitches used for this study. We show only high-amplitude glitches without the contribution of the Gaussian noise in order to highlight the temporal evolution of the transients. The detailed description of each glitch family is in section 4.1.

$$h(t) = h_0 \sin\left(2\pi f_0 (t - t_0)\right) \exp\left(-\frac{(t - t_0)^2}{2\tau^2}\right). \tag{7}$$

4.1.3. Ringdown (RD). These signals are short in time and have limited bandwidth, and can be modeled by damped sinusoids with parameters defined as for SG glitches. For times $t > t_0$ RD can be written as:

$$h(t) = h_0 \sin\left(2\pi f_0 (t - t_0)\right) \exp\left(-\frac{(t - t_0)^2}{2\tau}\right). \tag{8}$$

4.1.4. Chirp-like (CL). The chirping time evolution is very similar to the waveform produced by the coalescence of two compact objects such as neutron stars or black holes. Including this chirp-like family of glitches can show the potential of the method in discriminating noise from astrophysical signals, although the most robust way requires the comparison between the signals detected by different detectors [1]. This time evolution also reproduces the chirp-like observed in real data, that appears to be related to hardware injections [62]. We modeled these glitches using the parameters of the coalescence of a compact systems, i.e. the masses m_1 and m_2 of the components and the time of the coalescence t_0 as the time of the simulated glitch. From these parameters we can compute the chirp mass $\mathcal{M}_c = \frac{(m_1 m_2)^{3/5}}{(m_1+m_2)^{1/5}}$ and the the time to coalescence $\tau_c = t - t_0$, useful to express the chirp-like transients as:

$$h_+(t) = h_0 \tau_c^{-1/4} \cos\phi(t), \tag{9}$$

$$h_\times(t) = h_0 \tau_c^{-1/4} \sin\phi(t). \tag{10}$$

where the phase $\phi(t)$ is computed as:

$$\phi(t) = -2\left(\frac{5G\mathcal{M}_c}{c^3}\right)^{-\frac{5}{8}} \tau_c^{\frac{5}{8}} + \phi_0. \tag{11}$$

Without losing generality, for our simulations we fix $\phi_0 = 0$.

4.1.5. Scattered light-like (SL). These simulations aim at reproducing the glitches associated to scattered light in the interferometers. In the time-frequency plane, these glitches appear at low frequencies as a series of arcs that can last up to $\sim$1 s [62, 63]. We modeled this long-lasting glitches with a modified quadratic chirp centered at t_0 and modulated with a Gaussian of characteristic width τ. We simulate the glitch at a base frequency f_0 and its second and third harmonic.

$$h(t) = h_0 \sin(\phi_{\mathrm{SL}}) \exp\left(-\frac{(t-t_0)^2}{2\tau}\right). \tag{12}$$

Where the input ϕ_{SL} is defined as:

$$\phi_{\mathrm{SL}} = 2\pi f_0 (t - t_0)[1 - K(t - t_0)^2]. \tag{13}$$

We found that a value of $K = 0.5$ is good in reproducing the curvature of the scattered light glitches.

4.1.6. Whistle-like (WL). Whistles glitches have been observed in Advanced LIGO [62] and are related to MHz radio signals beating with the Voltage Controlled Oscillators. They occur at high frequencies and can extend for $\sim$0.5 s. We modeled them using the same functional form as scattered light, but fixing the curvature constant to a larger value related to the glitch width $K = 3\tau$, and removing its harmonics.

4.2. Simulated glitch sets

Using the glitch families described in the previous section we have built a simulated set to test the classification pipeline. Glitches are added to the simulation of Gaussian noise derived from the sensitivity curve of the LIGO H1 detector. The times t_0 at which the glitches occur are drawn from a Poisson distribution with mean rate of 0.5 Hz. With this choice it might be

that two glitches are nearby in time, thus offering the opportunity to test the classification performance and robustness against nearby glitches.

The sampling rate of the simulated time series is 8192 Hz, which is half the sampling rate used for LIGO and Virgo [64]. The original 16 kHz sampling rate could be used but will produce larger images and a large computational cost. We found that this choice is a good compromise between memory resources and the capability of highlighting fine details in glitches, that can be used for the classification. Without loss of generality, when processing real data we can simply downsample the measured $h(t)$ to the desired sampling rate.

The simulated glitch set has the purpose of testing the classification for glitches of different shapes in a wide range of duration, frequencies, and signal-to-noise ratios (SNR). We therefore included in this set 2000 glitches for each family described in section 4.1, plus an additional 2000 images from the NOISE class (see section 3.3) which adds up to a total of 14 000 samples. The summary of the parameters used for the simulations are reported in table 1.

In order to evaluate the accuracy of the algorithm as a function of the intensity of the transients, for each glitch we computed the optimal SNR based on matched filtering. The values of hrss have been chosen in order to have a maximum SNR of $\sim$60. The distribution of the SNR of the simulated glitches is shown in figure 4.

5. Results

5.1. Image preparation and processing

As anticipated in section 3.1, in order to build our training, validation, and test sample we have applied a whitening procedure to the simulated time series and then we have built the image spectrograms over a time window of 2 s centered on the glitch time. The size of the output images is 241 pixel along the time axis and 513 pixels along the frequency axis. We then applied clipping and contrast stretching as described in section 3.2.

In order to test the capability of the CNN to discriminate glitches from noise, we selected 2000 random windows of 2 s with no glitches and we built a seventh class, called NOISE, that we described in section 3.3 and added to our training, validation and test samples.

We also scaled the sizes of images by a factor of 0.55 on both axis. This scaling reduces the memory required without impacting significantly on the performance of the classification.

In order to show the impact of the whitening and image processing in cleaning the images, in figure 5 we show a gallery of spectrograms of simulated glitches without any processing. The images in this figure have been handpicked in order to better show the time-frequency charactestic evolution of each glitch family.

Furthermore, figure 6 shows the spectrograms of the same glitches of figure 5 after whitening and image processing steps have been applied. Although the whitening step removes most of the background noise, some residuals could be visibile at some frequencies, but they do not impact the classification task.

5.2. Detection and classification

We used the 15% of images as a test sample, 70% as training one and 15% as validation set. The training, validation and train set are stratified randomly selected.

As described in section 3.4, the output of the CNN is a vector of probabilities of glitches belonging to each of the 7 classes (6 glitch families and NOISE). In order to assign the class to each glitch, we take the class with the highest probability. This choice can be modified in

Table 1. Summary of the parameters used to prepare the simulated glitch set.

Glitch class	Parameter	Value
Gauss (GAUSS)	τ	4×10^{-4}–4×10^{-3} s
	hrss	2×10^{-23}–1.5×10^{-22} Hz$^{-0.5}$
sine-Gaussian (SG)	f_0	50–1500 Hz
	Q	2–20
	hrss	5×10^{-23}–1.5×10^{-22} Hz$^{-0.5}$
Chirp-like (CL)	M_1	1.4–30 $M_{\rm sol}$
	M_2	1.4–30 $M_{\rm sol}$
	hrss	7×10^{-23}–7×10^{-22} Hz$^{-0.5}$
Ringdown (RD)	f_0	50–1500 Hz
	Q	2–20
	hrss	5×10^{-23}–1.5×10^{-22} Hz$^{-0.5}$
Scattered light-like (SL)	f_0	32–64 Hz
	τ	0.2–0.5 s
	hrss	2×10^{-23}–2×10^{-22} Hz$^{-0.5}$
Whistle-like (WL)	f_0	500–3200 Hz
	τ	0.1–0.3 s
	hrss	5×10^{-23}–3×10^{-21} Hz$^{-0.5}$

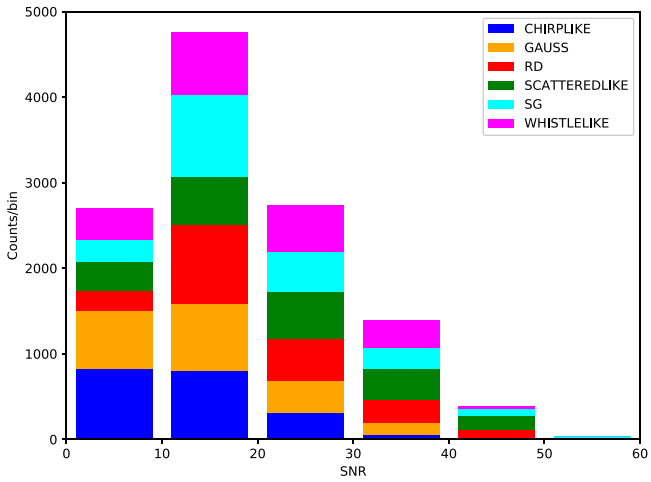


Figure 4. Distribution of SNR for the simulated glitch set.

order to have hints on glitch belonging to new, unknown, classes, e.g. by assigning the class only if the probability is above a certain threshold.

We have performed a tuning of the hyperparameters by studying the classification accuracy on the validation set. As an example, table 2 shows a scan in dropout strength, in order to find the values giving the highest accuracy. From this plot it can be seen that the maximum is between 0.2 and 0.3, then we decided to use 0.25 as our dropout strength for the deep network of figure 2. We also verified that for this value the accuracy still reaches the highest accuracy.

During the hyperparameter tuning we tested different batch sizes of 32, 64 and 128, and selected the value of 128 since it gives the lowest fluctuations in the training curve. The

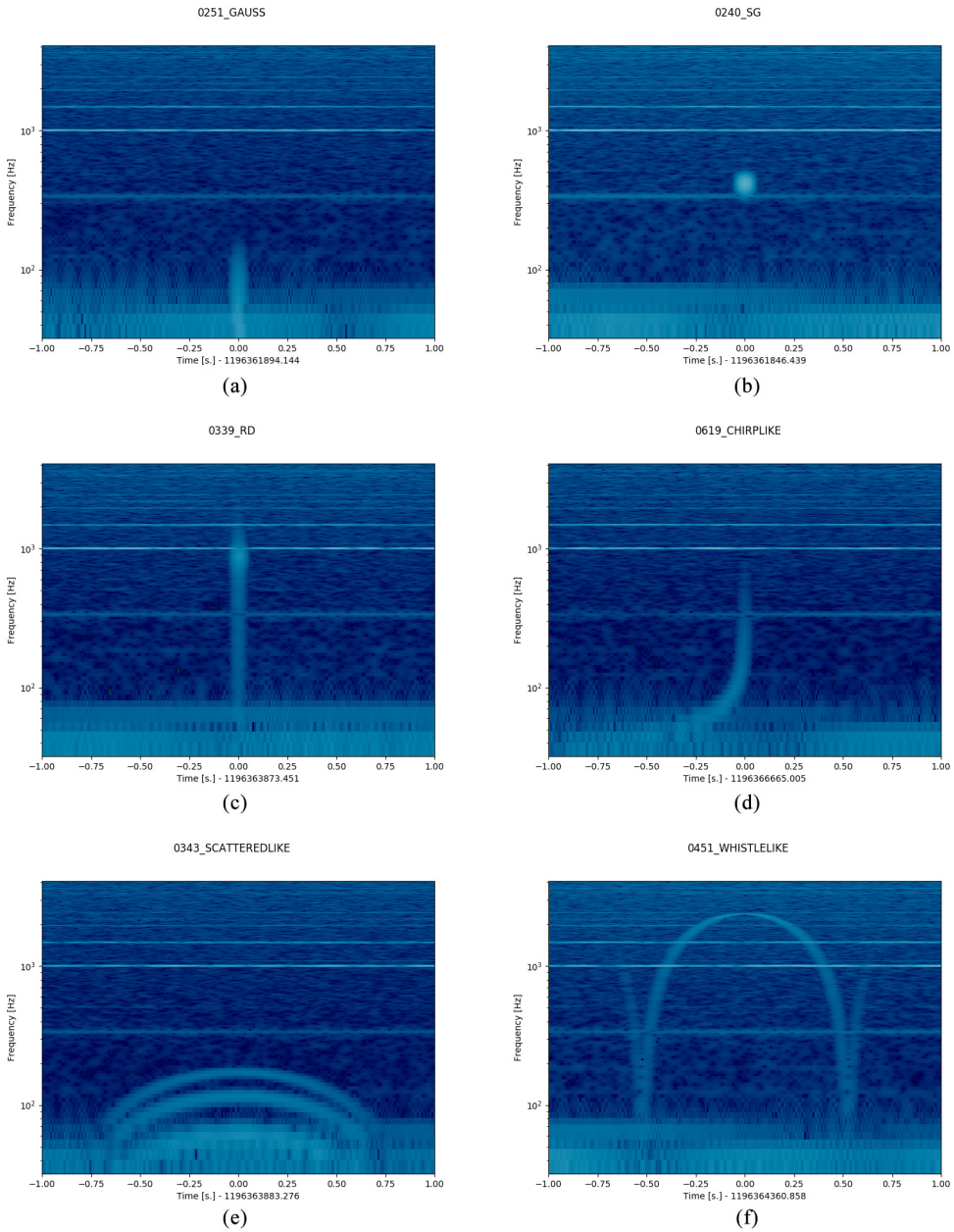


Figure 5. Gallery of sample spectrograms obtained using simulated glitches from simulated sets 1 and 2. (a): Gaussian (SNR ~ 7) (b): Sine Gaussian (SNR ~ 22) (c) Ringdown (SNR $= 12$) (d) Chirp-like ($\sim$8) (e) Scattered light-like (SNR ~ 24) (f) Whistle-like (SNR ~ 10). No whitening or image processing are applied.

accuracy has been computed using the validation set, and the optimal epoch has been chosen by looking at the evolution of the learning curve on the validation dataset. We have noticed that the accuracy is very high and having a further fine-tuning of the hyperparameters did not improve significantly more the accuracy.

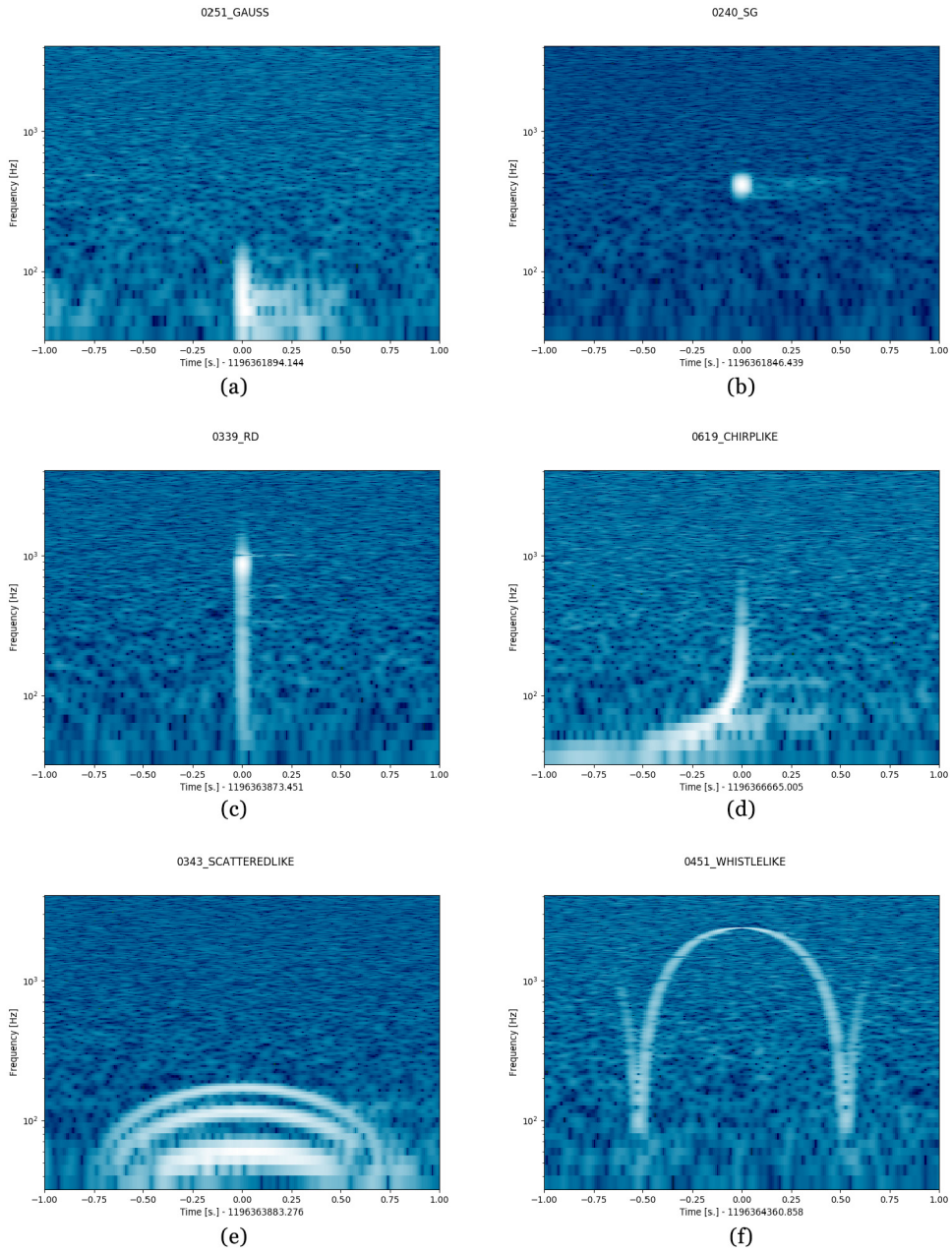


Figure 6. These spectrograms refer to the same glitches of figure 5, but here the time series have been whitened and the images are been further cleaned as explained in section 3.1.

We then studied how the architecture described in section 3.4 performed in comparison with simpler architectures. In particular, we considered a baseline machine learning algorithm (support vector machine) and two simpler neural networks. The size of images are limited by the memory of the GPU card, therefore for the classification using SVM the images have been

Table 2. The table shows how the accuracy of the validation dataset depends on the the dropout strength. The scan in dropout strenght has been performed on the simulated glitch dataset used for this work.

Dropout strength	Accuracy
0.1	0.9983
0.2	0.9987
0.3	0.9987
0.4	0.9986
0.5	0.9982
0.6	0.9947
0.7	0.9799
0.8	0.7594

reduced by a factor of 0.55, the same used for the CNNs. We used a linear SVM model and adopted a L2 regularization term in the cost function. We performed a search over the penalty parameter C and found 1.0 as the optimal value that maximizes the validation accuracy, estimated using a 10-fold cross validation procedure.

We then performed a comparison with a shallow network consisting of a single, 32-deep CNN hidden Keras Conv2D layer, followed by two fully connected Keras Dense layers with 64 and $N_c + 1$ neurons respectively, where N_c is the number of glitch classes and the additional one is inserted to represent the NOISE class. We also added to this comparison a more complex, two-layers network resembling the CNN blocks of our deep configuration described in section 3.4, made of two 16-deep Keras Conv2D layers followed by a Keras MaxPooling2D, a Keras dropout layer. The block is attached to two Keras Dense layers with 256 and $N_c + 1$ neurons respectively.

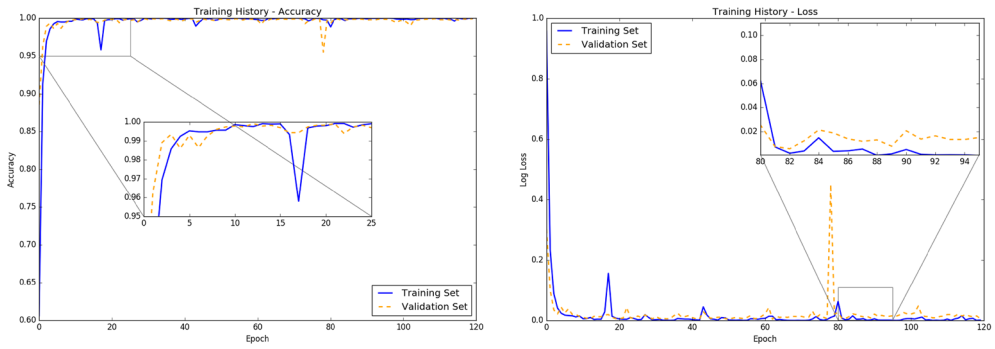
Table 3 reports the overall classification metrics for our deep network compared with other configurations, where it can be seen that a deeper network is performing better. In fact, although the baseline SVM algorithm is performing well on glitches that exhibit very different time-frequency behaviour, it has some difficulties in comparing glitches with similar shapes, such as GAUSS, SG and RD.

For the training phase of our 3-block network we used mini batches of 64 images. Figure 7 shows the training history curves as a function of training epoch, evaluated on the training and validation samples. The accuracy rises fast with the training epoch and reaches a plateau after $\sim$20 iterations. The evolution of the log loss of the validation sample is rapidly decreasing and starts rising significantly from $\sim$80 iterations, indicating the beginning of an overfitting regime. We therefore decided to stop our training after 70 epochs. The accuracy and log loss applied to the test sample is consistent with that obtained on the validation sample, showing that our network and its training procedure is reliable. Table 4 shows a summary of some important accuracy metrics used to evaluate machine learning classification algorithms, applied on the test set containing 15% of the data.

We then computed the confusion matrix for the classification, and the results are shown in figure 8 for both the SVM and the deep CNN classifier. We see that the accuracy on each class is greater than 99%. Both methods are excellent in distinguishing NOISE from glitches, but the CNN shows higher performance than SVM in distinguishing glitches with similar time-frequency behaviour such as GAUSS,SG and RD. Furthermore, it can happen that two glitches appear close in time and as a consequence their time-frequency evolutions will enter in the same image and make the classification a bit harder. However, the capability of CNNs to distinguishing features is making the algorithm robust against these cases. As an example,

Table 3. Comparison of performance between the selected deep architecture presented in section 3.4 and simpler CNN architecture. The metrics are evaluated on the images belonging to the validation sample.

Metric	Accuracy	Precision	Recall	F1 score	Log loss
SVM	0.971	0.972	0.971	0.971	0.08
Shallow CNN	0.986	0.986	0.986	0.986	0.04
1 CNN block	0.991	0.991	0.991	0.991	0.02
3 CNN blocks	0.998	0.998	0.998	0.998	0.008



(a) (b)

Figure 7. Evolution of accuracy (left) and log loss (right) as a function of the training epoch, applied to the training and validation sample. The insets show a zoom on the learning curve, showing respectively the regions where the accuracy reaches a plateau and where the log loss of the validation sample start to increase significantly.

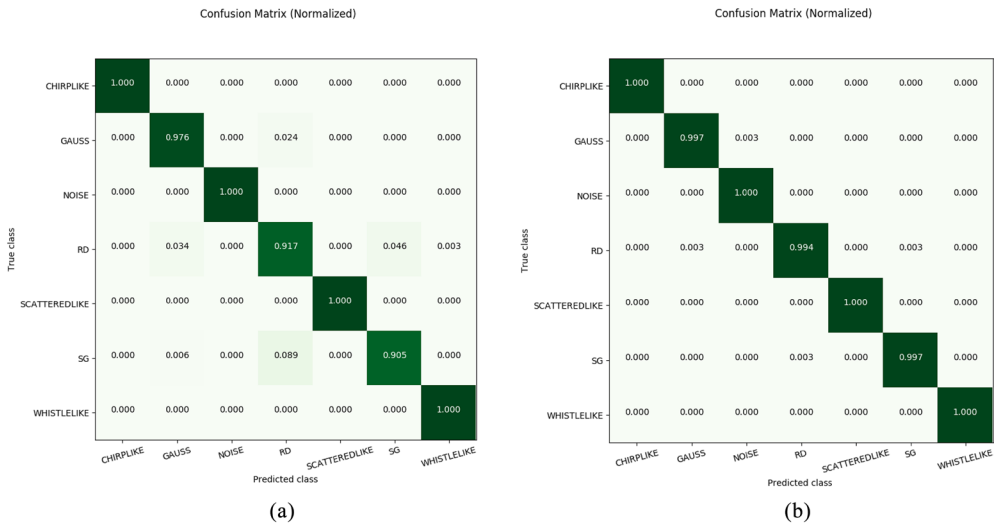
Table 4. Summary of metrics that describe the performance of the algorithm for the classification of the simulated glitch set. Each metric is evaluated on the test set.

Metric	Accuracy	Precision	Recall	F1 score	Log loss
ALL	0.998	0.998	0.998	0.998	0.006
GAUSS	0.997	0.997	0.997	0.997	0.008
SG	0.997	0.997	0.997	0.997	0.01
CHIRPLIKE	1.0	1.0	1.0	1.0	2×10^{-6}
RD	0.993	0.993	0.993	0.993	0.003
SCATTEREDLIKE	1.0	1.0	1.0	1.0	5×10^{-5}
WHISTLELIKE	1.0	1.0	1.0	1.0	3×10^{-8}
NOISE	1.0	1.0	1.0	1.0	5×10^{-7}

figure 9 shows a RD, SG and WHISTLELIKE glitch in the same image, and the algorithm computes the correct class (SG), since the output probability of belonging to that class is 1.0, while that of the other classes is $\sim$0.

6. Discussion

The results of the previous section show that image-based deep learning algorithms are very powerful in discriminating glitch transients from noise in gravitational wave interferometers.



(a) (b)

Figure 8. Confusion matrix for the multi-label classification based on images, evaluated on the test set.Left: baseline SVM classification algorithm. Right: CNN-based deep network described in section 3.4.

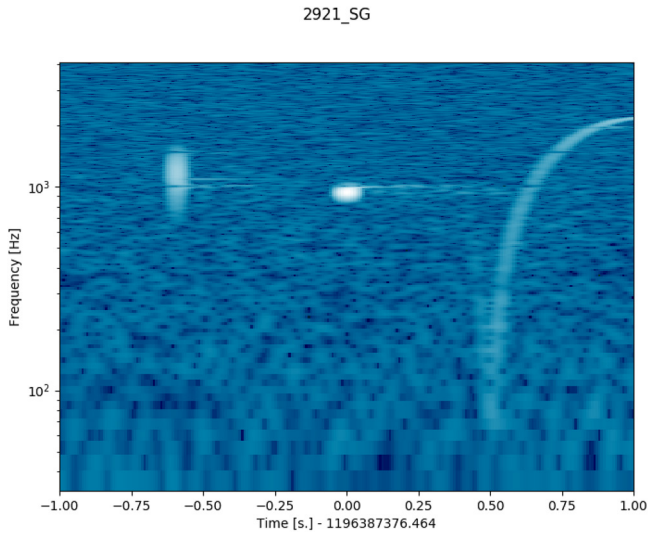


Figure 9. A more complex case of an image containing more than one glitch, namely a GAUSS, a SG and a WHISTLELIKE glitch.

Furthermore, the CNNs use the time-frequency dependence of the glitches to provide high accuracy in classification of glitches among different classes.

In order to reach high classification accuracy, in particular for low-SNR glitches, it is important to remove the contribution of the detector noise in the time series. Therefore, for our simulated glitches we applied a whitening procedure and an image processing stage, in order to enhance the visibility of the glitches in the images. In this way, $>99\%$ accuracy can be achieved.

Previous works have shown how CNN-based algorithms are performing better in classification tasks with respect to other machine learning algorithms such as support vector machines, logistic regression or random forest [31]. Furthermore, the capability of CNNs to extract features at various scales makes them more efficient with respect to the standard, fully-connected neural networks. This capability of extracting features in the images and use them is enhanced by some basic image processing steps.

In particular, we found that the capability of CNNs to look for features in the images has an important impact on the classification of time intervals containing glitches close to each other. Even in this case, the CNN was able to correctly recognize the glitch in the center of the image.

The duration of the training phase depends on the number of images in the training set, but this phase can be run on a dedicated GPU card, thus optimizing the resources. In our cases, the training of the architecture of section 3.4 over 70 epochs required about $\sim$5 h on a desktop PC equipped with a NVIDIA GeForce GTX 780 GPU card. Once the training is completed, the algorithm can classify each image within few microseconds. This approach is thus particularly important for low-latency applications, such as real-time, online recognition of glitches.

The parameters and weights of the neural network computed during the training phase can be stored on disk and loaded when necessary. For the dataset of 14 000 simulated glitches, the size of the model is $\sim$50 Mb. This is quite small, compared to the size of the spectrograms used ($\sim$13 Gb) and the spectrogram images ($\sim$2.4 Gb).

7. Conclusions

In this paper we have presented a new algorithm for the classification of transient noise glitches in advanced gravitational wave interferometers. The detection and identification of glitches are important tasks for the characterization of these complex detectors, in order to understand the origin of their noise sources and improve the sensitivity.

The algorithm has been built using convolutional neural networks, a class of deep learning algorithms very efficient for image processing and recognition. We have tested the proposed architecture against a set of simulated glitches superimposed to a colored Gaussian noise that reproduces to a good deal the actual noise in the advanced gravitational wave interferometers.

The presented results are encouraging and show the large potential of these algorithms for detector characterization. Without any particular need of optimization, the results show a $>$99% accuracy on the glitches simulated over a larger parameter space.

As noted also in previous works, this approach largely benefits from fast hardware available today, such as fast CPU and GPUs, as well as software developed for deep learning applications. This technology is intensively used in many fields and now is getting more and more important also in scientific research. In fact, the requirements in term of memory or disk space is within the reach of commercial hardware available today, making these algorithms very promising for various field of research.

Thanks to its flexibility, CNNs can be easily extended, e.g. to looking at new glitch families occurring in the detector and provide an automatic classification of them.

The goal of the algorithm is to provide an optimal automatic classification of glitches, but it could be customized to look for faint signals, e.g. from astrophysical sources [31]. The presented algorithm is able to process glitches in an autonomous way but can be combined with external pipelines for the detection of transient signals. We plan to run this algorithm on the real data collected by the Advanced LIGO and Advanced Virgo interferometers during observation runs. This CNN-based method is therefore very promising as a new, powerful tool

to investigate and characterize the transient noise in the era of advanced gravitational wave detectors.

Acknowledgments

We would like to thank the machine learning group and the detector characterization groups of the LIGO Scientific Collaboration and the Virgo Collaboration for the useful discussions of this work. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GeForce GTX 780 GPU used for this research. We also thank Marco Cavaglià for the internal LIGO-Virgo review and the anonymous referees for the insightful and useful comments and suggestions. This paper has been assigned the Virgo Document VIR-0669F-17 and LIGO Document P1700254.

ORCID iDs

Massimiliano Razzano https://orcid.org/0000-0003-4825-1629

References

[1] Abbott B P *et al* 2016 *Phys. Rev. Lett.* **116** 061102
[2] Abbott B P *et al* 2016 *Phys. Rev. Lett.* **116** 241103
[3] Abbott B P *et al* 2017 *Phys. Rev. Lett.* **118** 221101
[4] Abbott B P *et al* 2017 *Phys. Rev. Lett.* **119** 141101
[5] Abbott B P *et al* 2017 *Phys. Rev. Lett.* **119** 161101
[6] Gossan S E, Sutton P, Stuver A, Zanolin M, Gill K and Ott C D 2016 *Phys. Rev.* D **93** 042002
[7] Abbott B P *et al* 2017 *Astrophys. J.* **839** 12
[8] Aasi J *et al* (LIGO Scientific Collaboration) 2015 *Class. Quantum Grav.* **32** 074001
[9] Acernese F *et al* 2015 *Class. Quantum Grav.* **32** 024001
[10] Abbott B P *et al* 2016 *Living Rev. Relativ.* **19** 1
[11] Usman S A *et al* 2016 *Class. Quantum Grav.* **33** 215004
[12] Adams T, Buskulic D, Germain V, Guidi G M, Marion F, Montani M, Mours B, Piergiovanni F and Wang G 2016 *Class. Quantum Grav.* **33** 175012
[13] Klimenko S, Yakushin I, Mercer A and Mitselmakher G 2008 *Class. Quantum Grav.* **25** 114029
[14] Cella G and Giazotto A 2011 *Rev. Sci. Instrum.* **82** 101101
[15] Christensen N and (LIGO Scientific Collaboration and Virgo Collaboration) 2010 *Class. Quantum Grav.* **27** 194010
[16] Aasi J *et al* 2012 *Class. Quantum Grav.* **29** 155002
[17] Powell J, Trifirò D, Cuoco E, Heng I S and Cavaglià M 2015 *Class. Quantum Grav.* **32** 215012
[18] Calder A J, Burton A, Miller P, Young A W and Akamatsu S 2001 *Vis. Res.* **41** 1179–208
[19] Essick R, Vitale S, Katsavounidis E, Vedovato G and Klimenko S 2015 *Astrophys. J.* **800** 81
[20] Acernese F *et al* 2005 *Class. Quantum Grav.* **22** S1041
[21] Pedregosa F *et al* 2011 *J. Mach. Learn. Res.* **12** 2825–30 (http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf)
[22] Mukund N, Abraham S, Kandhasamy S, Mitra S and Philip N S 2017 *Phys. Rev.* D **95** 104059
[23] Samuel A L 1959 *IBM J. Res. Dev.* **3** 210–29
[24] Brockherde F, Vogt L, Li L, Tuckerman M E, Burke K and Müller K R 2017 *Nat. Commun.* **8** 872
[25] Li L, Baker T E, White S R and Burke K 2016 *Phys. Rev.* B **94** 245129
[26] Schütt K T, Arbabzadah F, Chmiela S, Müller K R and Tkatchenko A 2017 *Nat. Commun.* **8** 13890
[27] Kearnes S, McCloskey K, Berndl M, Pande V and Riley P 2016 *J. Comput. Aided Mol. Des.* **30** 595–608
[28] Faber F A, Hutchison L, Huang B, Gilmer J, Schoenholz S S, Dahl G E, Vinyals O, Kearnes S, Riley P F and von Lilienfeld O A 2017 *J. Chem. Theory Comput.* **13** 5255–64

[29] Alves A 2017 *J. Instrum.* **12** T05005
[30] Schmidhuber J 2015 *Neural Netw.* **61** 85–117
[31] George D and Huerta E A 2018 *Phys. Rev.* D **97** 044039
[32] Krizhevsky A, Sutskever I and Hinton G E 2012 *Proc. 25th Int. Conf. on Neural Information Processing Systems—Volume 1* (Red Hook, NY: Curran Associates Inc.) pp 1097–105
[33] Ioannidou A, Chatzilari E, Nikolopoulos S and Kompatsiaris I 2017 *ACM Comput. Surv.* **50** 1–38
[34] Druzhkov P N and Kustikova V D 2016 *Pattern Recognit. Image Anal.* **26** 9–15
[35] Zhang Z, Geiger J, Pohjalainen J, El-Desoky Mousa A, Jin W and Schuller B 2017 arXiv:1705.10874
[36] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K and Kuksa P 2011 *J. Mach. Learn. Res.* **12** 2493–537
[37] Lukic V and Brüggen M 2016 Galaxy classifications with deep learning *Proc. Int. Astronomical Union* **12** 217–20
[38] Schmelzle J, Lucchi A, Kacprzak T, Amara A, Sgier R, Réfrégier A and Hofmann T 2017 arXiv:1707.05167
[39] Hon M, Stello D and Yu J 2017 *Mon. Not. R. Astron. Soc.* **469** 4578–83
[40] McCulloch W S and Pitts W 1943 *Bull. Math. Biophys.* **5** 115–33
[41] Rosenblatt F 1958 *Psychol. Rev.* **65** 386–408
[42] Rumelhart D E, Hinton G E and Williams R J 1986 *Nature* **323** 533–6
[43] Hornik K, Stinchcombe M and White H 1989 *Neural Netw.* **2** 359–66
[44] Lecun Y, Bottou L, Bengio Y and Haffner P 1998 *Proc. IEEE* **86** 2278–324
[45] Hinton G E, Srivastava N, Krizhevsky A, Sutskever I and Salakhutdinov R R 2012 arXiv:1207.0580
[46] Rao P, Pereira N A and Srinivasan R 2016 *2nd Int. Conf. on Contemporary Computing and Informatics* pp 489–93
[47] Flamary R 2016 arXiv:1612.04526
[48] George D, Shen H and Huerta E A 2017 arXiv:1706.07446
[49] Cuoco E, Losurdo G, Calamai G, Fabbroni L, Mazzoni M, Stanga R, Guidi G and Vetrano F 2001 *Phys. Rev.* D **64** 122002
[50] Cuoco E, Cella G and Guidi G M 2004 *Class. Quantum Grav.* **21** S801–6
[51] Kay S M 1988 *Modern Spectral Estimation: Theory and Application/Book and Disk (Prentice-Hall Signal Processing Series)* (Englewood Cliffs, NJ: Prentice Hall)
[52] Acernese F *et al* 2007 *Class. Quantum Grav.* **24** S671–9
[53] Cuoco E, Calamai G, Fabbroni L, Losurdo G, Mazzoni M, Stanga R and Vetrano F 2001 *Class. Quantum Grav.* **18** 1727–51
[54] Bahaadini S, Rohani N, Coughlin S, Zevin M, Kalogera V and Katsaggelos A K 2017 arXiv: 1705.00034
[55] Krizhevsky A, Sutskever I and Hinton G E 2012 ImageNet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems 25 (NIPS 2012)* ed F Pereira *et al* pp 1097–105
[56] van der Walt S, Colbert S C and Varoquaux G 2011 *Comput. Sci. Eng.* **13** 22–30
[57] Oliphant T E 2007 *Comput. Sci. Eng.* **9** 10–20
[58] Theano Development Team 2016 arXiv:1605.02688
[59] Chollet F *et al* 2015 https://github.com/fchollet/keras
[60] Chetlur S, Woolley C, Vandermersch P, Cohen J, Tran J, Catanzaro B and Shelhamer E 2014 arXiv:1410.0759
[61] Abbott B P *et al* 2009 *Phys. Rev.* D **80** 102001
[62] Zevin M *et al* 2017 *Class. Quantum Grav.* **34** 064003
[63] Vinet J Y, Brisson V and Braccini S 1996 *Phys. Rev.* D **54** 1276–86
[64] Abbott B P *et al* 2016 *Class. Quantum Grav.* **33** 134001