Università degli Studi di Padova

Department of Physics and Astronomy "Galileo Galilei"

Master degree in Astrophysics and Cosmology

Final dissertation

Search for gravitational wave signals in LIGO-Virgo-KAGRA data using machine learning methods

Supervisor

Prof. Marco Bazzan

Università degli Studi di Padova

Co-supervisor

Prof. Agata Trovato

Prof. Edoardo Milotti

Università degli Studi di Trieste

Candidate

Francesco Quaglio

Student ID

2028638

Academic Year

2024-2025

— Ai miei genitori Annarosa e Roberto —

Abstract

The aim of this work is to explore the possibility of using neural network classifiers to identify gravitational-wave signals in the LIGO-Virgo-KAGRA data. This task is complicated by the non-Gaussian, non-stationary noise in the detectors, in particular in periods when only one detector takes data. One detrimental source of noise is the so-called glitches, non-Gaussian transient noise that can mimic astrophysical signals. In general, temporal coincidence between two or more detectors is used to mitigate contamination, but when a single detector is in operation, coincidence is impossible. In this context, machine learning strategies could help complement more traditional approaches.

One problem often faced in machine learning applications is the need for a large amount of training data. It is therefore useful to have a way to simulate glitches and gravitational-wave signals. While techniques for simulating gravitational-wave signals are well established, the simulation of glitches is less common because of their variability and unknown origin. In this work, a conjecture about the possible nature of some glitch classes is used to simulate them.

Contents

List of figures

List of tables

Glossary of acronyms

ASD . . . . . amplitude spectral density

AUC . . . . . area under the curve

BBH . . . . . binary black hole

BH . . . . . black hole

BNS . . . . . binary neutron star

BS . . . . . beam splitter

CBC . . . . . compact binary coalescence

CNN . . . . . convolutional neural network

EFE . . . . . Einstein field equation

XGBoost . . . . . eXtreme Gradient Boosting

GPS . . . . . Global Positioning System

GR . . . . . general relativity

GW . . . . . gravitational wave

GWOSC . . . . . Gravitational Wave Open Science Center

IT . . . . . InceptionTime

LVK . . . . . LIGO, Virgo, KAGRA

MAE . . . . . mean absolute error

ML . . . . . machine learning

NN . . . . . neural network

NR . . . . . numerical relativity

PDF . . . . . probability density function

PSD . . . . . power spectral density

ROC . . . . . receiver operating characteristic

SNR . . . . . signal-to-noise ratio

STFT . . . . . short-time Fourier transform

TT . . . . . transverse traceless

Introduction

The detection of the first gravitational wave (GW) on 14 September 2015, announced by the LIGO and Virgo Collaborations, opened a completely new way to study astrophysical phenomena [11]. This signal, known as GW150914, originated from the merger of a binary black hole (BBH) system and was identified using matched-filtering techniques. Its amplitude is sufficiently large to be clearly visible in the data even by eye [12]. Matched filtering is a method used to identify GW signals, which are almost always hidden within the noise. This technique operates by comparing the detector data with simulated waveforms representing the expected GW signal. Although computationally expensive, it remains the standard approach for GW detection. The data used to detect GW signals come from the strain measurements recorded by interferometers such as LIGO, Virgo, and KAGRA (LVK), and represent relative changes in the arm lengths induced by passing GWs. The first detection took place during the O1 observing run, followed by O2, O3 and O4. During the O2 run, for the first time, a binary neutron star (BNS) merger was detected, further enriching the catalog of GW sources [13]. Other potential sources include core-collapse supernovae, the collapse of massive stars to black holes (BH), and even relic signals from the Big Bang, but so far none of these have been observed [14, 15].

The LVK collaborations have publicly released the interferometric strain data collected from O1 up to the first part of O4. All the detected events are cataloged on the GWTC releases [8, 16, 17, 1, 7]. In particular, the latest GWTC–4.0 has been made available, including 128 newly identified GW candidates.

The strain data from interferometers are affected by noise. In addition to Gaussian and stationary noise, they also contain short-duration noise artifacts, known as 'glitches' [18]. These glitches can mimic the signal of a GW [19], creating confusion in the data and complicating the identification of GW events. In general, astrophysical signals can be distinguished from this type of noise because they appear in temporal coincidence across multiple detectors [20]. However, random temporal coincidences between glitches are still detrimental and the situation is even worse when only one detector is operating and this strategy cannot be applied. In fact, during the observing runs, the detectors are not always active simultaneously, and in those specific periods the possibility of detecting new GW signals is limited.

Understanding glitches is a common challenge in GW data analysis and is essential for devel-

oping alternative strategies for GW detection. Glitches show large variability in both the duration and the frequency range [19]. In addition, the existence of many different glitch classes further complicates the construction of models capable of identifying them from the data [20]. In order to develop these models, it is essential to be able to simulate glitches. However, doing so requires understanding their physical origin, which is largely unknown. Among the various classes identified, a subset of glitches shares similar morphological features and may originate from the same physical mechanism: Barkhausen noise [10]. Based on this assumption, it becomes possible to simulate this type of non-Gaussian noise.

Being able to simulate these glitches opens up several possible applications. One of these is the use of simulated glitches to enrich the training datasets of a neural network (NN) classifier capable of distinguishing instrumental noise, glitches, and GW signals. Studies of this kind are becoming increasingly necessary, given the continuous growth in detector sensitivity and the consequently growing rate of detected events. The development of faster, complementary methods will be essential to support traditional techniques in the identification of GWs during both single-detector and multi-detector observation periods. In this work, starting from a Barkhausen model, glitches are simulated and injected into the strain data, focusing in particular on the LIGO Livingston detector data from O3b. To make the simulations consistent with real observations, the generation parameters are estimated with the machine learning method eXtreme Gradient Boosting (XGBoost) [21], using the glitch features defined in [10]. This thesis then employs one of the NN classifiers used in [20], training it on time-series datasets composed of instrumental noise, glitches, and simulated GW signals, in order to evaluate its performance when including the glitches generated in this study.

The first chapter (1) introduces GWs, starting from the general concept of waves and gravity in general relativity, and illustrating them through the solutions of Einstein's linearized equations. Particular attention is given to how GWs affect spacetime, how they are generated in the case of compact binary coalescences (CBCs), and how they can be observed.

The second chapter (2) focuses on the analysis of data collected by the LVK network. It presents standard methods for extracting GW signals from noisy data, with particular attention to the characterization of the detector noise and data quality.

The third chapter (3) presents the machine learning (ML) approaches used in this work for GW data. Both classification and regression tasks are discussed, including the design and implementation of XGBoost and NNs.

Chapter (4) focuses on the characterization and simulation of a particular class of glitches. It first presents how they are classified and then introduces a method to simulate specific classes

using a model based on Barkhausen noise. Finally, the chapter describes an ML approach based on XGBoost to estimate the characteristic parameters of this model from the features of real glitches.

Chapter (5) presents the results obtained from a NN classifier trained on noise, glitches, and GW signals, highlighting its performance and ability to distinguish among the three classes.

The final chapter summarizes the conclusions, presents the main findings, and outlines possible directions for future work.

1 Gravitational waves

Gravitational waves (GWs) represent one of the most important discoveries in modern physics. They opened a new window for the study of some of the most energetic phenomena in the universe. Understanding the mechanisms by which they are generated and their observable effects is essential to develop methods to detect them.

This chapter begins by reviewing the general concept of waves in physics. Then, it focuses on GWs, discussing their nature, starting from the fundamental principles of general relativity (GR). In addition, it introduces ground-based detectors, such as LIGO, Virgo, and KAGRA, outlining how they work to detect GWs.

The final part of the chapter describes binary black hole (BBH) mergers, a particular astrophysical system that emits gravitational radiation. In particular, it focuses on the evolution of the GW signal during the final stages of the system.

1.1 Waves in physics

The content of this section follows [22]. In physics, a wave is commonly described as a perturbation of an equilibrium state that propagates with a well-defined velocity.

In nature, we observe many types of waves: sound waves, which propagate through gases such as air; seismic waves, which travel through the ground; and surface waves on a liquid, like sea waves. All of these are examples of mechanical waves that require a medium to propagate. In such waves, the perturbation travels through the oscillatory motion of atoms and molecules

around their equilibrium positions. They transport energy and momentum, but not matter. However, other waves do not require a medium to propagate. Electromagnetic waves, for example, are oscillations of electric and magnetic fields, which allow them to travel even through vacuum.

The origin of a wave is called its source. In the case of mechanical waves, the source is typically a vibration of a material body; for electromagnetic waves, the source could be the acceleration of an electric charge.

Like electromagnetic waves, they do not need a medium to propagate, but are instead perturbations of space and time themselves, generated by changes in the distribution of mass. These are known as gravitational waves, whose existence was first predicted by the theory of general relativity. To better understand their nature and origin, it is necessary to take a step back and clarify what gravity means.

1.2 Gravitational waves: basic concepts

As mentioned earlier, when discussing GWs, we referred to mass, a concept that plays a central role in Newton's theory of universal gravitation. Following the description of Newtonian gravity in [23], two masses attract each other with a force whose magnitude is proportional to the product of their masses and inversely proportional to the square of the distance between them. This theory has been confirmed by numerous observations in astronomy. However, in 1915, one notable anomaly remained: the unexplained motion of Mercury. Since it is the closest planet to the Sun, Mercury is subject to an intense gravitational field, and in such regimes, Newtonian gravity shows its limitations.

1.2.1 A new view of gravity

The unexpected motion of Mercury is explained by Einstein's formulation of gravity. Following the description in [24], in GR gravity is not described as a new force or as a new classical field. What really changes is our understanding of space and time. Einstein defines gravity as the curvature of spacetime: the four-dimensional union of space and time. According to GR, mass and energy deform spacetime, and objects move in response to this curvature. For example, the Sun does not exert a force on the Earth, but his mass curves the spacetime around it and so the Earth follows this geometry.

In a schematic form, this idea is summarized by the following relation:

$$(\text{a measure of spacetime curvature}) = (\text{a measure of matter energy density})$$

In a more formal way, Einstein's field equations (EFEs) are written as follows[1]:

$$\boldsymbol{R_{\mu\nu}} - \frac{1}{2}\boldsymbol{R g_{\mu\nu}} = 8\pi \boldsymbol{G T_{\mu\nu}} \tag{1.1}$$

This is a fundamental equation in physics and it cannot be derived from more basic principles. As seen in the previous schematic form, the left-hand side represents the curvature of spacetime, expressed through the Ricci tensor and the Ricci scalar. The right-hand side contains the stress-energy tensor which describes the distribution of matter and energy.

Another key element in the equation is the symmetric tensor $\boldsymbol{g_{\mu\nu}}$, called the metric, which plays a fundamental role in describing the geometry of spacetime and is essential for understanding the nature of GWs. This tensor defines the line element

$$\boldsymbol{ds^2} = \boldsymbol{g_{\mu\nu}}(\boldsymbol{x})\, \boldsymbol{dx^\mu dx^\nu}$$

which is used to compute the spacetime distance between two events separated by coordinate differences $\boldsymbol{dx^\mu}$. In flat spacetime, with no curvature, the metric reduces to the Minkowski metric $\boldsymbol{\eta_{\mu\nu}}$. Here, flat refers to the three-dimensional Euclidean space, assumed in Newtonian mechanics, with the addition of a time component.

1.2.2 Linearized EFEs in vacuum

The EFEs are a set of nonlinear partial differential equations and there is no method to find a "general solution". Only particular solutions are given under specific assumptions. One such case is the linearized form of the EFEs in vacuum. In vacuum $\boldsymbol{T_{\mu\nu}}$ vanishes and, after some manipulation, the EFEs reduce to

$$\boldsymbol{R_{\mu\nu}} = 0$$

By inserting a small perturbation around the Minkowski metric

$$\boldsymbol{g_{\mu\nu}}(\boldsymbol{x}) = \boldsymbol{\eta_{\mu\nu}} + \boldsymbol{h_{\mu\nu}}(\boldsymbol{x}) \tag{1.2}$$

[1] Using natural units where the speed of light is set to unity.

into the Ricci tensor, and expanding the vacuum EFEs to first order in $h_{\mu\nu}(x)$, one obtains the linearized field equations. These equations can be simplified by an appropriate choice of coordinates. One possible approach is to consider a small and arbitrary translation of the coordinates x^{μ} of the same order as $h_{\mu\nu}(x)$. Then, by choosing an appropriate gauge condition (the Lorenz gauge), the linearized EFEs in vacuum reduce to a set of simple wave equations:

$$\boxed{\Box\, h_{\mu\nu}(x) = 0}$$

It follows that the solutions of the linearized EFEs in vacuum describe a small perturbation of the Minkowski metric that propagates at the speed of light. Such solutions are interpreted as the propagation of weak GWs through an empty and flat spacetime.

The description of GW solutions below is based on [24] and [25]. The Lorenz gauge does not completely fix the gauge freedom and a residual freedom remains. An appropriate choice of coordinates can be used to simplify the wave equations: this is known as the transverse-traceless (TT) gauge. Starting from sixteen degrees of freedom, the number reduces to ten due to the symmetry of the metric perturbation tensor. By applying the residual gauge freedom, only two physical degrees of freedom remain, which correspond to the plus (h_{+}) and the cross ($h_{\times}$) polarizations of GWs.

The solution is a plane wave of the form:

$$h^{TT}_{\mu\nu} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_{+} & h_{\times} & 0 \\ 0 & h_{\times} & -h_{+} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} e^{i\omega\,(t-z)}$$

assuming the wave propagates along the z-axis with frequency ω. This solution is also referred to as the waveform of the GW.

Since the waveform has no components along the longitudinal direction, the GW does not affect the separation between test masses aligned along the z-axis. Instead, only the transverse separations in the xy-plane oscillate in time as the wave passes.

To better understand the physical effect of GWs, one approach is to observe the time behavior of a ring of free test particles, initially at rest in the transverse xy-plane of the wave. Focusing on the plus polarization component, as shown in table 1.1, the ring of particles oscillates stretching and squeezing along transverse directions under the effect of the GW. Additionally, the table 1.1

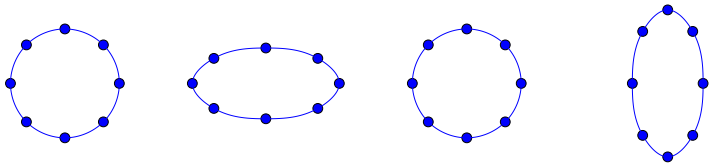
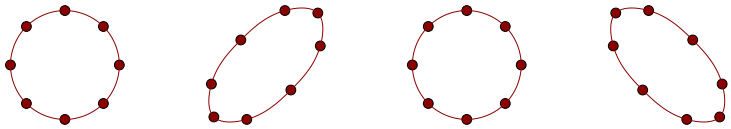
ωt	0	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$
h_+				
$h_\times$				

Table 1.1: The table shows the physical effect of a GW on a ring of free test particles lying in the plane transverse to its direction of propagation. The waveform is computed at a fixed spatial location along the direction of propagation (e.g. $z = 0$), with the origin of time chosen such that $h_{\mu\nu}$ vanishes at $t = 0$. Top: effect of the h_+ polarization. Bottom: effect of the $h_\times$ polarization.

clearly shows that the deformation of the ring due to the cross polarization is the same of the plus polarization but rotated by 45°.

Summarizing, from the linearized EFEs in vacuum, GWs:

- are valid only in the regime where the perturbation is very small;
- propagate at the speed of light;
- are transverse waves;
- have two independent polarization states;
- carry energy and momentum [2], as their effect is to induce motion in free test particles through periodic stretching and squeezing their relative positions.

Consequently, the detection of a GW does not require the observation of the motion of a single particle, but the relative displacement between at least two test masses. But how small is the displacement to be measured? Quantitatively, the typical amplitude of h is $\sim 10^{-21}$ [11]. In terms of physical displacement, for test particles separated by a distance of about one kilometer, this corresponds to variations of the order $\sim 10^{-18}$ meters, i.e. 200 times smaller than the radius of a proton [9].

The next section shows how such incredibly small displacements can be detected.

[2] Without going into technical details, GWs should not be think only as perturbations traveling through spacetime. Since they carry energy, they can themselves act as sources of spacetime curvature. This goes beyond the assumption of a flat background metric.

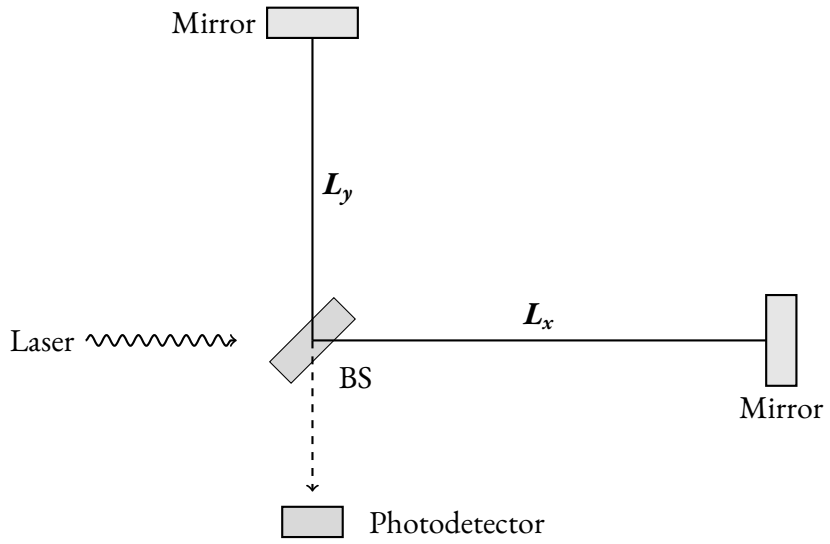


Figure 1.1: Schematic diagram of a Michelson interferometer. A laser beam hits the BS, which splits the beam into the two perpendicular arms of the interferometer, denoted $\boldsymbol{L_x}$ and $\boldsymbol{L_y}$. The beams are reflected by the mirrors at the end of the arms and then return to the BS, where they are recombined and directed toward the photodetector.

1.3 Ground-based detectors

As shown in the previous section, two test particles subjected to a GW will start oscillating around their initial positions, as the wave stretches and squeezes the distance between them. To detect this effect, the main idea is to use a Michelson interferometer. The discussion of interferometers in the following is based on [24]. In practice, three test masses are placed in free fall, so that all other forces acting on them are minimized. Two of these test masses are mirrors, while the third is a beam splitter (BS).

As illustrated in figure 1.1, an incident laser beam is split by the BS into two perpendicular directions. These directions are the arms of the interferometer and are indicated as $\boldsymbol{L_x}$ and $\boldsymbol{L_y}$. The beams are reflected by the mirrors at the end of the arms and then return to the BS, where they are recombined and directed toward the photodetector. When the two beams are recombined, they can interfere constructively or destructively. This interference depends directly on the path length difference between the two arms[3]. As the difference varies, the intensity of the light measured by the photodetector also changes. This principle is the core of GW detection. So an interferometric GW detector translates a metric perturbation into a measurable

[3] More precisely, what the interferometer measures is not the difference in the physical lengths of the arms, but the difference in the time it takes for light to complete a round trip along each arm. This time delay is observed as the phase difference between the two beams when they recombine.

signal [9].

Although the Michelson interferometer is conceptually simple, real GW detectors are significantly more complex. To maximize the response of the interferometer to the passage of a GW, it can be shown that the optimal arm length is approximately

$$L \simeq 750 \text{ km} \left(\frac{100 \text{ Hz}}{f_{\text{gw}}} \right)$$

where f_{gw} is the frequency of the wave. For typical frequencies around 100 Hz, this implies optimal arms of several hundred kilometers. However, such dimensions are impractical for ground-based detectors due to both technical and financial constraints. In practice, current ground-based detectors use much shorter arms (only a few kilometers). These interferometers are designed to extend the optical path of their arms without physically increasing their length. This is achieved by inserting Fabry–Pérot cavities into the arms: within these cavities, the laser beams bounce multiple times between highly reflective mirrors before being recombined by the BS. For this to occur, servo systems are used to keep the laser in resonance [9].

In addition to these, several optical systems are placed both between the laser source and the BS, and between the BS and the photodetector, in order to increase the laser power, stabilize the laser frequency [9].

To effectively keep the mirrors in free fall, they must remain suspended and as stationary as possible. This is achieved using a multistage pendulum system called the "superattenuator", which acts as chains of harmonic oscillators [9].

Nowadays, GW observations are enabled by a network of interferometers, including LIGO, Virgo, and KAGRA (LVK). In chapter 2, we will explore how to extract and analyze data from public catalogs.

Noise budget of GW detectors

A critical aspect of detecting GW is the study of the different noise components and how to minimize them. The output of any GW detector is a combination of a GW signal and noise [25]. The strain noise spectrum of the detectors in figure 1.2 shows that the interferometer response to displacement is frequency dependent, and for LVK network the best sensitivity is between 100 Hz and 300 Hz [9]. The figure shows the overall strain sensitivity, resulting from the sum of the contributions of the various noise sources. In addition, the strain noise spectrum of LVK network exhibits several narrow "lines". These lines are carefully studied to

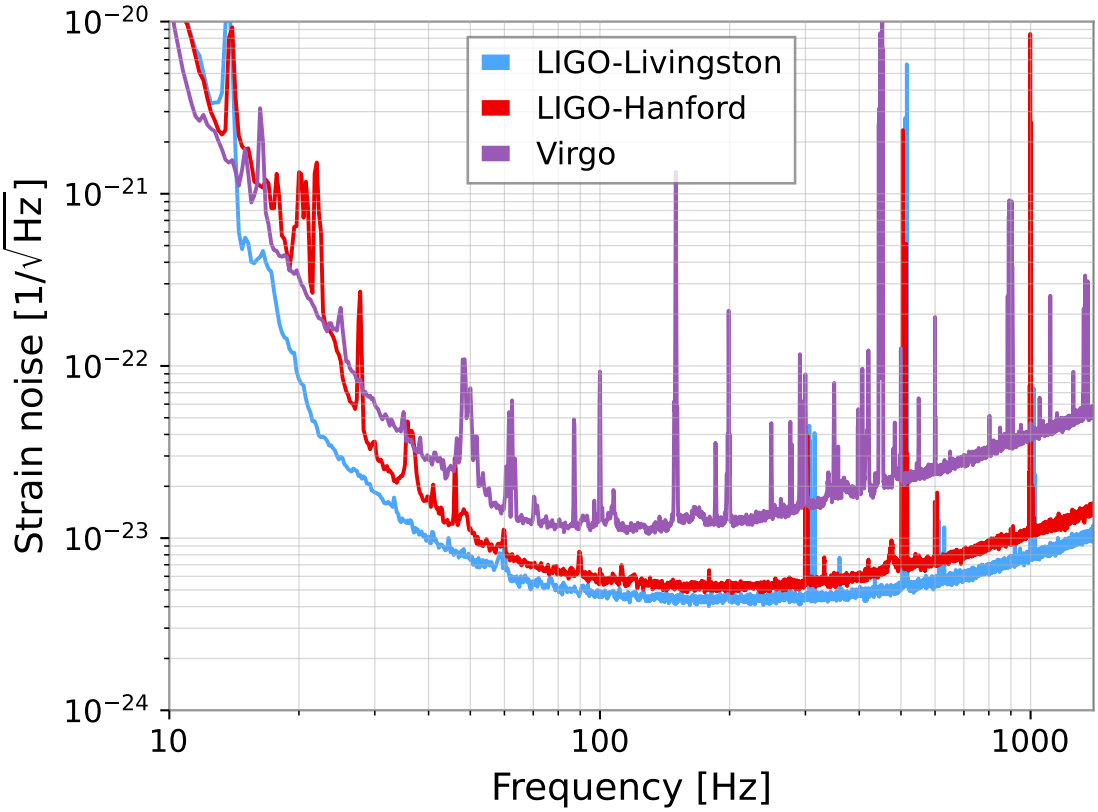


Figure 1.2: Amplitude spectral density (ASD) of the strain noise for the LIGO Hanford, the LIGO Livingston, and the Virgo detectors during O3, computed around the time of the GW190412 event [1] using the `asd()` function of GWPy [2].

understand their origins, and the detectors are designed to keep their bandwidth as narrow as possible [26].

Starting from low frequencies, the main challenge is to prevent the mirrors from moving [9]. The first important source of noise is seismic noise. This noise originates from seismic activity, human activity, or ocean waves, and can push the mirrors out of alignment. It dominates at low frequencies, typically in the 1–10 Hz range. To reduce this effect, active stabilization is required [9]. As seen previously, the mirrors are suspended in a system of harmonic oscillators and each pendulum acts as a high-pass filter, attenuating frequencies above its resonance frequency. The lower the resonance frequency, the better the isolation from seismic noise [25]. Another source of misalignment arises from thermally driven motion of both the suspension system and the mirrors, as well as from mechanical losses in the mirror coatings known as thermal noise. Dielectric multilayer coatings are designed to ensure high reflectivity while minimizing optical absorption. Thermal noise limits the sensitivity of the interferometers around 60 Hz [9]. Test masses are also sensitive to variations in the local gravitational field caused by changes in nearby mass distributions. Although this is not a limiting factor for current detectors, at design sensitivity this time-varying gravitational field, known as Newtonian noise, could become important below about 20 Hz [9]. Even with improved seismic and thermal isolation, Newtonian noise sets the ultimate low-frequency limit for ground-based detectors [25]. At high frequencies, the sensitivity of the detector is ultimately limited by quantum noise, which includes shot noise and radiation pressure noise [9]. Shot noise arises from the discrete nature of photons and their Poisson-distributed arrival times that dominates at high frequencies. At the same time, the radiation pressure from photons on the mirrors introduces low frequency noise [9]. Shot noise decreases with the inverse square root of the laser power circulating in the interferometer arms [25] and is limiting above 70 Hz [9]. During the first observing runs, LIGO operated with about 100 kW of circulating laser power, but it can reach up to 750 kW to further reduce shot noise. However, since higher power also increases radiation pressure noise, a compromise between the two effects is required [25].

The noise sources described so far generate a Gaussian and stationary background. However, additional non-Gaussian and transient noise, known as glitches, is added on this background.

Over time, several studies have attempted to morphologically classify glitches into ***families***; a well-known example is Gravity Spy [4], which categorizes them based on their time–frequency maps. As shown in figure 1.3, glitches span a wide range of frequencies and typically last no more than a few seconds.

The classes differ significantly, with some being louder than others, occurring predomi-

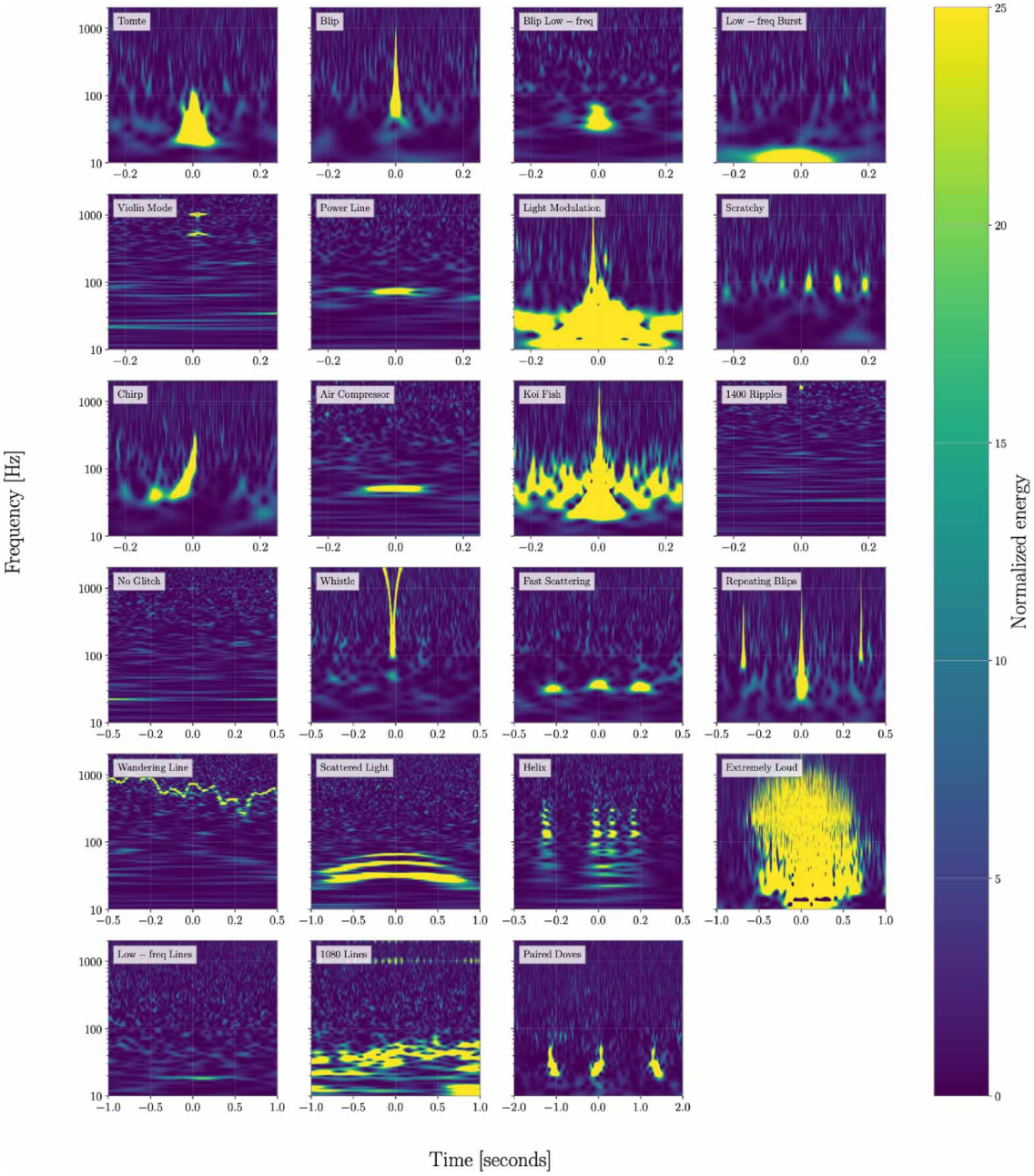


Figure 1.3: Time–frequency map of glitches, adapted from [3], classified by Gravity Spy [4]. The glitches span a wide range of frequencies and durations. The different classes are distinct, with some being louder than others, some occur predominantly at low or high frequencies, and some have very short or longer durations.

nantly at low or high frequencies, or having very short or longer durations. The physical origin of glitches remains largely unknown, although correlations have been observed between the increased observation rate of scattered light and fast scattering, and the ground motion due to earthquake or human activity [3]. Moreover, since detectors such as LIGO, Virgo, and KAGRA contain numerous electronic and electromagnetic components, these can act as sources of unwanted signals. For example, the superattenuators include coils, sensors, and control systems used to keep the system aligned. All these elements can introduce small disturbances or electromagnetic interference which, once propagated to the detector output, appear as non-Gaussian noise. These effects often arise from complex and chaotic processes that are difficult to model, and can generate glitches that contaminate the measurement [10]. The main glitch classes that could be related to such processes will be examined in detail in chapter 4.

Having seen how GW detectors work, the next question is what generates GWs and what their signals look like.

1.4 Astrophysical sources of GWs

The previous sections discussed what GWs are and how they can be observed. In section 1.1, it was mentioned that the origin of a perturbation is referred to as a source. But what exactly gives rise to these ripples in spacetime curvature?

As described in [24], to discuss GW sources, the first step is to return to the EFEs and study their solutions when the stress energy tensor does not vanish (see equation 1.1). In this context, GW sources are associated with non-zero components of $\boldsymbol{T}_{\mu\nu}$. EFEs are solved by assuming that GWs are produced by a weak source, so that, as in the vacuum case, the resulting metric perturbation is small and can be treated as a perturbation around a flat spacetime (see equation 1.2). In such conditions, linearized Einstein equation (in Lorentz gauge) can be written as follows

$$\boxed{\Box \boldsymbol{h}_{\mu\nu}(\boldsymbol{x}) = -16\pi \boldsymbol{T}_{\mu\nu}}$$

Although these equations admit exact solutions, in this context the solution can be found under a set of approximations: the long wavelength limit, where the wavelength of the GW is much larger than the characteristic size of the source; the large distance approximation, where the distance from the source is much greater than its radius; and a weak source limit. This last approximation refers to the case in which velocities within the source are small compared to the speed of light. Such sources are typically referred to as non-relativistic. This approximation

allows for a multipole expansion on powers of the typical velocity inside the source. The leading term in the multipole expansion is the second time derivative of the mass quadrupole moment[4], which is related to the dynamic behavior of the source's mass distribution. Since the emission is proportional to the nonuniform variation of the quadrupole mass moment, it is directly related to the asymmetry of the source. Consequently, non-symmetric sources, such as rotating stars that are not perfectly symmetric with respect to their rotation axis, binary systems, or supernovae exploding asymmetrically, can emit GWs.
In this approximation, moreover, the solution depends inversely on the distance between the observer and the source, and is evaluated at the retarded time to account for the delay between the emission of the GW and its detection.

1.4.1 Compact binary coalescence

The first direct detection of GWs by ground-based interferometers was produced by a BBH system with masses of 36 $M_\odot$ and 29 $M_\odot$ at a distance from the Earth of about 410 Mpc [9]. The observation mainly involves the inspiral and the coalescence of the BHs. The observed signal shows both an increase in amplitude and frequency, which can only be explained by an inspiraling motion of two bodies with a decreasing orbital radius [12]. At a certain point, the rising frequency terminates, and this can be explained with the end of the inspiral phase. After this, the amplitude decreases and the frequency stabilizes, indicating that the system has reached a new stable equilibrium. The only plausible explanation for this behavior is the merger of the two compact objects [12].

This thesis focuses on the analysis of this particular type of signal, known as compact binary coalescence (CBC) signals. To understand its nature, one must examine the solutions of the wave equation in the case of such a system. The following description is based on [25].

To describe a GW observed from a binary system in circular orbit a useful mass quantity is called chirp mass and is defined as follows

$$\mathcal{M} = \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}}$$

where m_1 and m_2 are the masses of the two compact objects.

At sufficiently low velocities and large orbital radius, Newtonian dynamics describes very

[4] More precisely, the mass quadrupole moment is defined as $M^{ij} = \int d^3x \rho(t, x) x^i x^j$ where ρ is the rest mass density.

well the circular motion of two objects [12]. Using Kepler's law, the orbital radius can be expressed as a function of the binary's orbital frequency. However, this frequency is not constant: because of the energy loss via GW emission, the orbit gradually shrinks, which increases the orbital frequency.

However, as the frequency increases, the power radiated in GWs also increases. This establishes a back-reaction mechanism that, over a sufficiently long timescale, leads to the coalescence of the binary system.

The GW frequency [5] can be expressed in terms of the chirp mass of the system and the time remaining before the merger measured by the observer τ [6]. The GW frequency is then given by

$$f_{\mathrm{gw}}(\tau) \simeq 134\,\mathrm{Hz}\left(\frac{1.21\,M_{\odot}}{\mathcal{M}}\right)^{5/8}\left(\frac{1\,\mathrm{s}}{\tau}\right)^{3/8} \tag{1.3}$$

As can be seen, from the measurement of the GW frequency and τ [7], it is possible to determine the chirp mass and thus obtain a combination of the masses of the two objects.

Under the standard approximations of a weak source, long wavelength, and large distance, the linearized EFEs in the TT gauge admit only two independent degrees of freedom. The solution for a binary system can then be written in terms of the time remaining before the coalescence, as follows

$$\begin{cases} h_{+}(\tau) = \frac{1}{r}(G\mathcal{M})^{5/4}(\frac{5}{\tau})^{1/4}(\frac{1+\cos^2\theta}{2})\cos(\Phi(\tau)) \\ h_{\times}(\tau) = \frac{1}{r}(G\mathcal{M})^{5/4}(\frac{5}{\tau})^{1/4}\cos\theta\sin(\Phi(\tau)) \end{cases} \tag{1.4}$$

where r is the distance between the source and the observer, and $\Phi(\tau)$ is the phase function of the waveform. In the simplified case where the GW frequency is constant, the phase evolves linearly with time. However, in a binary system, the frequency increases as the system approaches coalescence, due to the energy loss via GW emission. In this case, $\Phi(\tau)$ evolves approximately as $(-\tau/\mathcal{M})^{5/8}$ and the GW amplitude and GW frequency grow as τ decreases. This behavior is referred to as a chirp because of its similarity to the sound emitted by a bird. For this reason, the resulting signal is known as a chirping waveform.

[5] It is equal to twice the orbital frequency of the binary system because the quadrupole moment is invariant under a rotation by π around the orbital axis.

[6] The time before the coalescence is defined as $t_{\mathrm{coal}} - t$. Note that this definition implicitly omits the concept of retarded time. However, since τ is defined as the difference between two times, and the retarded time differs from the observer time by a constant, it follows that $(t_{\mathrm{coal}})_{\mathrm{ret}} - t_{\mathrm{ret}} = t_{\mathrm{coal}} - t$.

[7] Or, in a similar approach, by measuring the GW frequency and its variation over time.

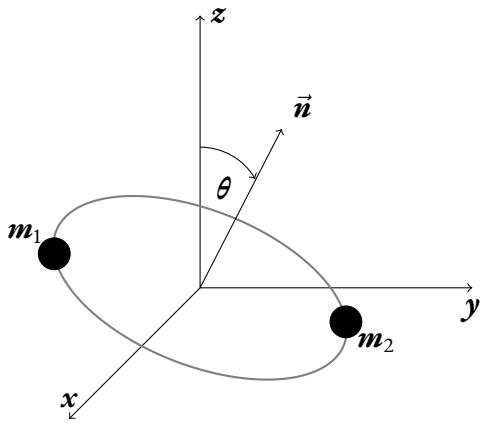


Figure 1.4: The figure shows a BBH system with component masses m_1 and m_2 orbiting in a plane orthogonal to the vector $\vec{n}$, which forms an angle θ with respect to the z-axis. This orbital motion generates GWs that can be detected from any direction. Without loss of generality, the line of sight is oriented along the z-axis.

The dependence on the angle θ can be understood by looking at figure 1.4. It represents a binary system with masses m_1 and m_2 orbiting in a plane orthogonal to the normal vector $\vec{n}$. For an observer placed along the z-axis the GW propagates in that direction. θ is defined as the angle between the line of sight and the rotation axis of the system. When the system is edge-on respect the observer, the cross polarization vanishes. In this case, the wave is linearly polarized. On the other hand, when the system is face-on, the two polarizations have the same amplitude, resulting in a circularly polarized wave. Therefore, the degree of polarization carries information about the inclination of the orbital plane.

1.4.2 Revisiting the assumptions

The equations above 1.3 and 1.4 describe the behavior of a binary system during the inspiral phase with GW emission. A more general approach that includes the non linear effect of the EFE is a series of expansions called post-Newtonian (PN) approximation. This approach expresses the system's energy and the flux irradiated as a power series in the orbital velocity, assuming it remains small compared to the speed of light [27]. The higher order of this expansion takes into account several physical effects not present in the lower order. These effects, for example, consider self-spin and spin-spin coupling. In fact, during orbital motion, the possible spins of the two objects should also be taken into account. This modifies their gravitational radii and their orbital dynamics [12].

So far, in this description of the motion of two compact objects, the orbit is assumed as

circular. However, this is not the general case. In fact, the orbit of two compact objects is elliptical, and Kepler's law applies to the semi-major axis. This reduces the chirp mass observed in 1.3. However, since GWs carry away angular momentum, the orbit circularizes very rapidly, much faster than the rate at which it shrinks, so relaxing this assumption does not significantly affect the final outcome [12].

Moreover, GWs are affected by the expansion of the universe, which stretches their frequency for an observer on Earth. So, due to cosmological redshift, the masses in the detector frame appear larger than the masses for the source frame by a factor of $(1+z)$ where z is the redshift [12].

1.4.3 Beyond the inspiral phase

As can be seen, equations 1.3 and 1.4 are valid during the inspiral phase, i.e., for times before coalescence. However, beyond this phase, it is clear that a different description is required. The same consideration applies to the PN approximation, which breaks down as the velocities involved become comparable to the speed of light. In this regime the solution of EFEs can be approximated only by numerical relativity (NR) [27]. It corresponds to the merger of the two BHs and the formation of the final remnant, defined only by its mass and spin [12]. As described in [28], the remnant is a Kerr BH with a mass equal to the sum of the individual BH masses minus the energy radiated through GWs emission. For example, in the case of GW150914, the energy radiated corresponds to approximately 3 $M_\odot$ and the final BH has a mass of about 62 $M_\odot$ [9]. The waveform after coalescence is called the ringdown and is produced by the oscillations of the remnant BH. During this phase, the signal consists of a superposition of damped sinusoids and so an exponential decay in amplitude [28].

1.4.4 CBC waveform

In summary, the evolution of the BBH system can be divided into three distinct phases: (i) inspiral, (ii) coalescence, and (iii) ringdown. An illustration of these three phases is shown at the top of figure 1.5.

The inspiral phase is characterized by a quasi-circular orbit of the binary, with velocities much smaller than the speed of light [27]. In the case of an eccentric binary, the emission of GWs quickly circularizes the orbit.

The coalescence phase corresponds to the merger of the two BHs, during which their velocities become comparable to, or exceed, half the speed of light [27].

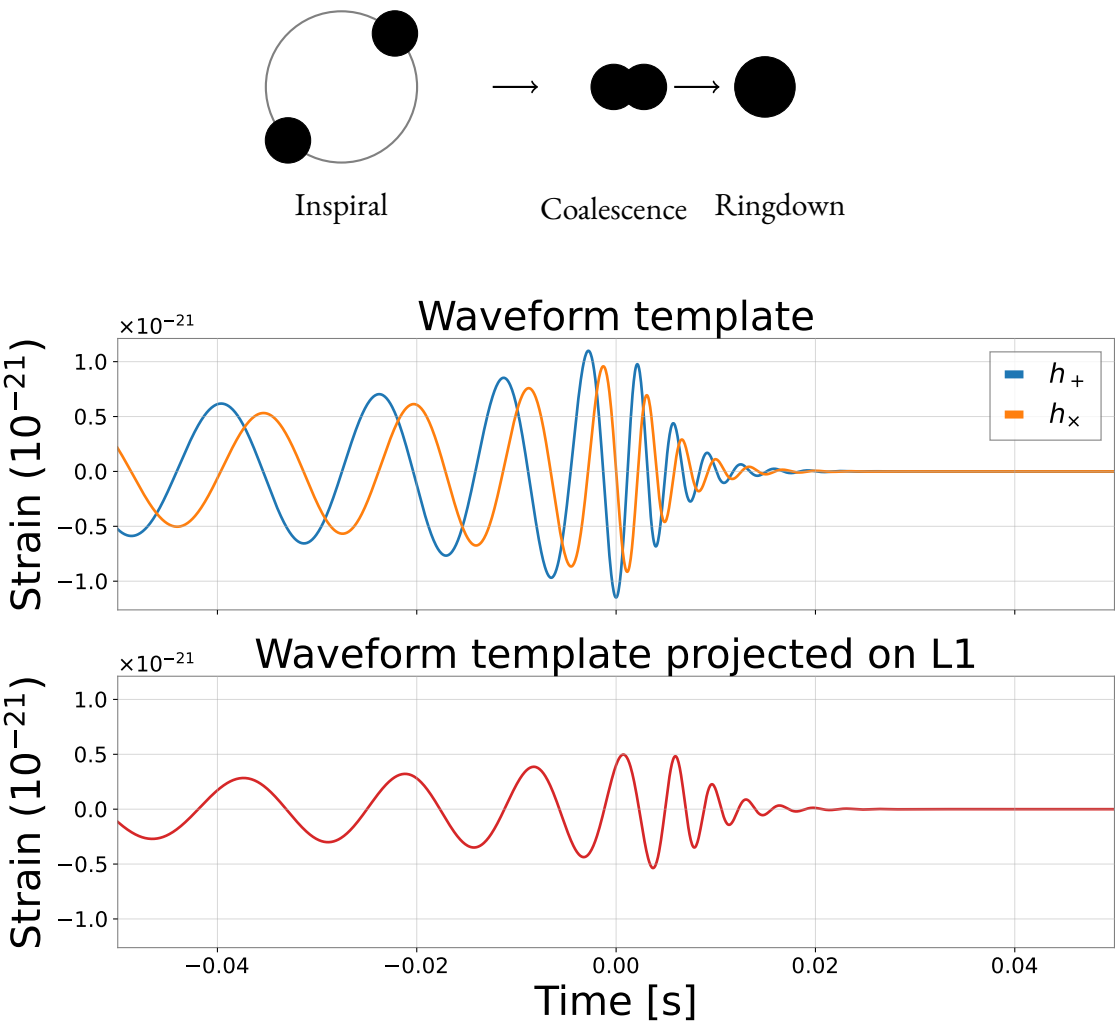


Figure 1.5: Top: Evolution of the BBH system in three phases: (i) inspiral, (ii) coalescence, and (iii) ringdown. Middle: NR simulation of a CBC signal obtained with the SEOBNRv4 [5] waveform model implemented in PyCBC [6], showing both h_+ and $h_\times$ polarizations. The source masses are set to $30\,M_\odot$, with zero spins. The inclination is 1 rad and the luminosity distance is 400 Mpc. Bottom: Projection of the waveform onto the LIGO Livingston detector using example values for sky location and polarization: $\alpha = 1.4$ rad, $\delta = -1.2$ rad, and $\psi = 0$ rad.

Finally, during the ringdown phase, the final remnant radiates away its excess energy in the form of GWs.

As shown in figure 1.5, the numerically computed waveform exhibits a characteristic chirp signal during the inspiral (with increasing amplitude and frequency), reaches a peak during coalescence, and is followed by an exponential decay in amplitude during last phase. These simulations can describe with high precision GWs emitted by CBC [27].

To simulate a GW signal produced by a CBC, it is necessary to know the parameters describing the source. Some of these describe the intrinsic properties of the source, such as the masses of the two components, and the pair of spins projected along the orbital angular momentum axis. In addition, some extrinsic parameters are required to generate the waveform, such as the inclination of orbital angular momentum with respect to the line of sight, the luminosity distance to the observer, and the phase at coalescence [27]. These parameters determine the two polarizations of the wave, $\boldsymbol{h}_+$ and $\boldsymbol{h}_\times$.

The detector response, $\boldsymbol{h}(\boldsymbol{t})$, is a combination of the two polarizations. However, the signal is also affected by the orientation of the detector relative to the wave direction. This dependence is described by the so-called antenna pattern, and the signal observed by the detector can be written as:

$$\boldsymbol{h}(\boldsymbol{t}) = \boldsymbol{F}_+(\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\psi})\, \boldsymbol{h}_+(\boldsymbol{t}) + \boldsymbol{F}_\times(\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\psi})\, \boldsymbol{h}_\times(\boldsymbol{t})$$

where $\boldsymbol{F}_+$ and $\boldsymbol{F}_\times$ are the detector pattern functions, which depend on the sky location of the source (right ascension $\boldsymbol{\alpha}$ and declination $\boldsymbol{\delta}$), and the polarization angle $\boldsymbol{\psi}$. The polarization angle defines the rotation of the two polarizations within the plane orthogonal to the direction of propagation of the GW. Consequently, to obtain the detector response, the two polarizations must be projected through the detector pattern functions [27].

A CBC waveform can be simulated using `get_td_waveform()` from the PyCBC package [6] and one of the available waveform models, as shown in figure 1.5. The inclination angle ranges from 0 to $\boldsymbol{\pi}$, while the coalescence phase varies between 0 and $2\boldsymbol{\pi}$. The right ascension $\boldsymbol{\alpha}$ and the polarization phase $\boldsymbol{\psi}$ are defined over $[0, 2\boldsymbol{\pi}]$, whereas the declination $\boldsymbol{\delta}$ spans from $[-\boldsymbol{\pi}/2, \boldsymbol{\pi}/2]$. In the simulated signal shown in figure 1.5, the two polarizations $\boldsymbol{h}_+$ and $\boldsymbol{h}_\times$ differ in amplitude due to the inclination of the source relative to the line of sight. The lower panel of the figure shows the response of the LIGO Livingston detector, obtained by projecting both polarizations with the `project_wave()` function according to the source's sky-location parameters.

2

Detection and data analysis

The detection of gravitational waves (GWs) requires not only sophisticated instruments, which transform the phenomenon into measurable data, but also detailed analysis to extract information from them.

This chapter focuses on the practical aspects of working with GW data. It explains how to extract data from the GWOSC archive, discusses the impact of noise, examines data-quality categories, and presents the main tools used to identify GW events.

2.1 Public data from GWOSC

LIGO, Virgo, and KAGRA constitute a network of interferometers designed to observe GWs. Their data and analysis products are publicly released through the Gravitational Wave Open Science Center (GWOSC) [1]. Publicly available data include the strain time series recorded by the LVK ground-based detectors, as well as additional channels reporting the instrument conditions and segments that specify the quality of the data [18]. Moreover, the catalogs include lists of detected events and corresponding parameter estimates.

The LVK network operates during different observation runs, each corresponding to specific periods. The completed runs are listed below, together with their Global Positioning System (GPS) time intervals [18]:

[1] GWOSC home page: `https://gwosc.org`

- O1: from 12 September 2015 (GPS 1126051217) to 19 January 2016 (GPS 1137254417)
- O2: from 30 November 2016 (GPS 1164556817) to 25 August 2017 (GPS 1187733618)
- O3: from 1 April 2019 (GPS 1238166018) to 27 March 2020 (GPS 1269363618)
- O4a: from 24 May 2023 (GPS 1368975618) to 16 January 2024 (GPS 1389456018)

During the observation runs, the interferometers do not record data continuously. Each detector experiences interruptions due to maintenance, seismic disturbances, or other technical issues [18]. Consequently, the instruments are not always operating simultaneously, and there are periods when only one detector is active. Consequently, the strain data are divided into segments of varying lengths.

The interferometers produce time series representing the fractional difference in length between their two arms. The raw data recorded by the photodetector, which measures the laser light at the output of the interferometer, are converted into an accurate measurement of the gravitational strain $\boldsymbol{h}(\boldsymbol{t})$ through a calibration procedure. This procedure relies on a detector response model that accounts for all sources of motion in the differential arm cavity length, including environmental contributions [29].

The strain data of LVK can be downloaded from the GWOSC website or via the `fetch_open_data()` command of the GWPy package [2]. In the default version of these data, they are provided with the best available calibration. This method allows selecting both the detector from which the data are downloaded and the time interval of interest.

The LVK network operates at a nominal sampling rate of 16384 Hz, but the data available for download are provided both at the original sampling frequency and at a reduced rate of 4096 Hz [18]. Additional downsampling can be performed depending on the type of analysis.

2.2 Noise and data quality

Most of the data recorded by the detectors consist of instrumental noise. Although, as a first approximation, this noise can be considered stationary and Gaussian, in reality it often deviates from these assumptions. Studying its characteristics is therefore essential to assess the quality of the data. Its properties are influenced by the factors described in section 1.3, in particular the combination of seismic, thermal, quantum, and Newtonian noise.

2.2.1 Characterization of the detector noise

Characterizing this noise is a crucial step for GW data analysis, since the detectability of a GW signal strongly depends on the noise spectrum of the interferometer.

A common way to study the noise properties of a detector is through the estimation of its power spectral density (PSD) $\boldsymbol{S_n(f)}$, which quantifies how the power of the signal is distributed as a function of frequency.

In other words, the PSD allows one to identify the frequency bands where the instrument is most sensitive or dominated by noise.

It is often more convenient to represent the PSD in terms of amplitude by taking the square root of the power, obtaining the Amplitude Spectral Density (ASD) shows in figure 1.2. The ASD was computed using the `asd()` method of the `GWPy` library [2], which implements Welch's technique by combining the Fourier transforms of the time series calculated on overlapping, windowed segments of data [30].

Having the ASD allows us to whiten the data, meaning that the strain can be rescaled according to the detector sensitivity across different frequencies. In practice, this means giving less weight to frequencies where the detector is less sensitive and more weight to the frequency range where the sensitivity is higher, typically around 100 Hz and 300 Hz. This process facilitates the identification of GW signals. Mathematically, whitening consists of three steps [31]:

- computing the Fourier transform of the strain to obtain the data in the frequency domain;
- dividing the spectrum by the ASD;
- performing the inverse Fourier transform to recover the whitened strain in the time domain.

$$\boldsymbol{d(t)} \xrightarrow{\text{FT}} \boldsymbol{\tilde{d}(f)} \xrightarrow{\text{Whitening}} \boldsymbol{\tilde{d}_w(f) = \frac{\tilde{d}(f)}{\sqrt{S_n(f)}}} \xrightarrow{\text{iFT}} \boldsymbol{d_w(t)}$$

In addition, after the whitening, a band-pass filter is used in GW data pre-processing to isolate the frequency range where most astrophysical signals are expected. Limiting the data to a band helps suppress low-frequency and high-frequency noise, focusing the analysis on the most informative spectral region. Figure 2.1 shows the results of the procedure described above, applied to a 4-second segment of data from the latest O4 run around the candidate GW250114, as listed in the GWTC−4 catalog [7]. The figure illustrates both the whitening and band-pass filtering steps.

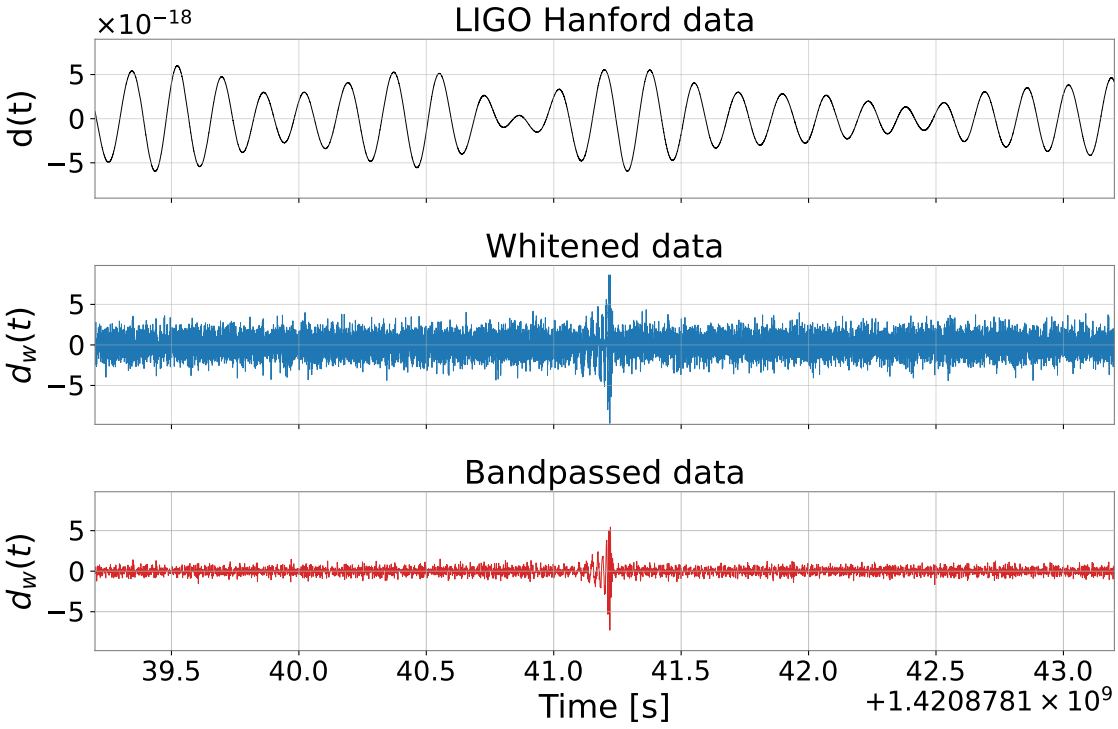


Figure 2.1: Top: Calibrated strain from the LIGO Hanford detector around GW250114 [7], shown for a 4-second interval centered at GPS time 1420878141.2. Middle: The data are whitened using the ASD to account for the detector noise. Bottom: A bandpass filter between 30 Hz and 500 Hz is applied to highlight the relevant signal components.

For this purpose, 200 s of data around the GW250114 event were downloaded. The time series was then whitened using the `whiten()` function and bandpass-filtered with the `bandpass()` function, both from the GWPy package [2].

2.2.2 Non-stationary noise and data quality

As mentioned previously, the noise recorded by GW detectors is not truly stationary and Gaussian. The data are affected by instrumental and environmental artifacts, which can appear in time-series data as short, non Gaussian transients, known as glitches [18].

GW detectors can experience longer periods of reduced performance caused by issues in the control systems, calibration procedures, or environmental effects. Such conditions can lower the overall detector sensitivity [18]. The LVK collaboration employs several data-quality metrics to detect and mitigate these effects in the data and uses data-quality flags to label the segments [18] as follows. The most severe ones are labeled as CAT1 and correspond to times when the detector was not operating properly. These segments are excluded from all searches and parameter estimation analyses [18]. CAT2 flags indicate less critical or shorter issues, while CAT3 flags identify possible anomalies of unknown origin [18].

Time–frequency representation of GW data

GW scientists often use time-frequency representations to visually inspect the data. For example, a commonly used tool for the detection of short, unmodeled transients is the short-time Fourier transform (STFT). This method applies Fourier transforms to short, windowed segments of the signal, and the resolution in time and frequency depends on the chosen window length [32]. However, a single fixed window cannot represent all frequencies with same resolution. Low-frequency signals require long windows to achieve sufficient frequency resolution, whereas high-frequency signals require shorter windows to have good temporal localization [32].

To address this limitation, the Q-transform is an extension of the standard STFT in which the length of the analysis window changes inversely with frequency [32].The window length is controlled by the quality factor Q, which determines the trade-off between time and frequency resolution, as shown in figure 2.2.

Given a signal $\boldsymbol{x}(\boldsymbol{t})$, the Q-transform is defined as follows [33]:

$$\boldsymbol{X}(\boldsymbol{\tau}, \boldsymbol{f}, \boldsymbol{Q}) = \int_{-\infty}^{+\infty} \boldsymbol{x}(\boldsymbol{t})\, \boldsymbol{w}(\boldsymbol{t} - \boldsymbol{\tau}, \boldsymbol{f}, \boldsymbol{Q})\, \boldsymbol{e}^{-2\pi i \boldsymbol{f} \boldsymbol{t}}\, \boldsymbol{dt}$$

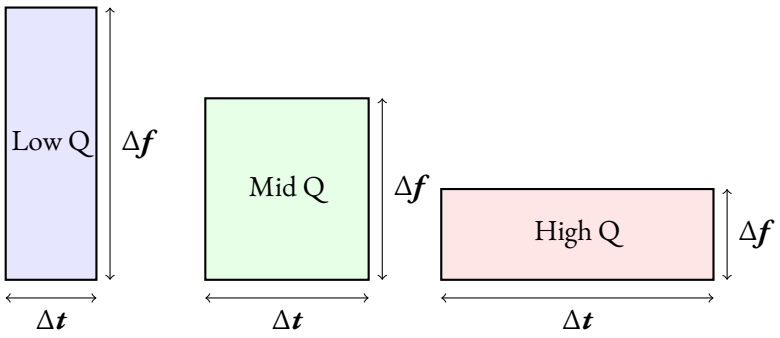


Figure 2.2: Illustration of the adaptive time-frequency tiling used in the Q-transform. Low-Q components correspond to long time windows and narrow frequency bands, while high-Q components correspond to short time windows and wide frequency bands.

Here, $w(t-\tau, f, Q)$ is a time-domain window centered at τ, whose duration is determined by f and the chosen Q factor.

Figure 2.3 shows examples of the time–frequency representation for the first GW detection and for a glitch, obtained using the `q_transform()` function from GWPy [2]. This implementation of the Q-transform algorithm automatically selects the optimal Q value.

2.3 Classical detection methods

The GWOSC Event Portal is an online platform that provides access to a catalog of published GW transient events. For each event, the portal presents a variety of associated data. These include the strain data of the event, measures of detection confidence, and estimates of astrophysical source parameters. Most of them are signals from CBCs, although some may originate from instrumental noise [18].

In GW searches, candidate events are initially identified using the matched-filtering technique. These triggers indicate the presence of a potential signal and the following section explains how it works.

2.3.1 Matched filtering for CBC signals

As previously mentioned, GW detectors record strain data as discrete time series. A segment of data can be written as

$$d = \{d(t_1), \ldots, d(t_N)\}, \quad \text{with } N = T \cdot f_s$$

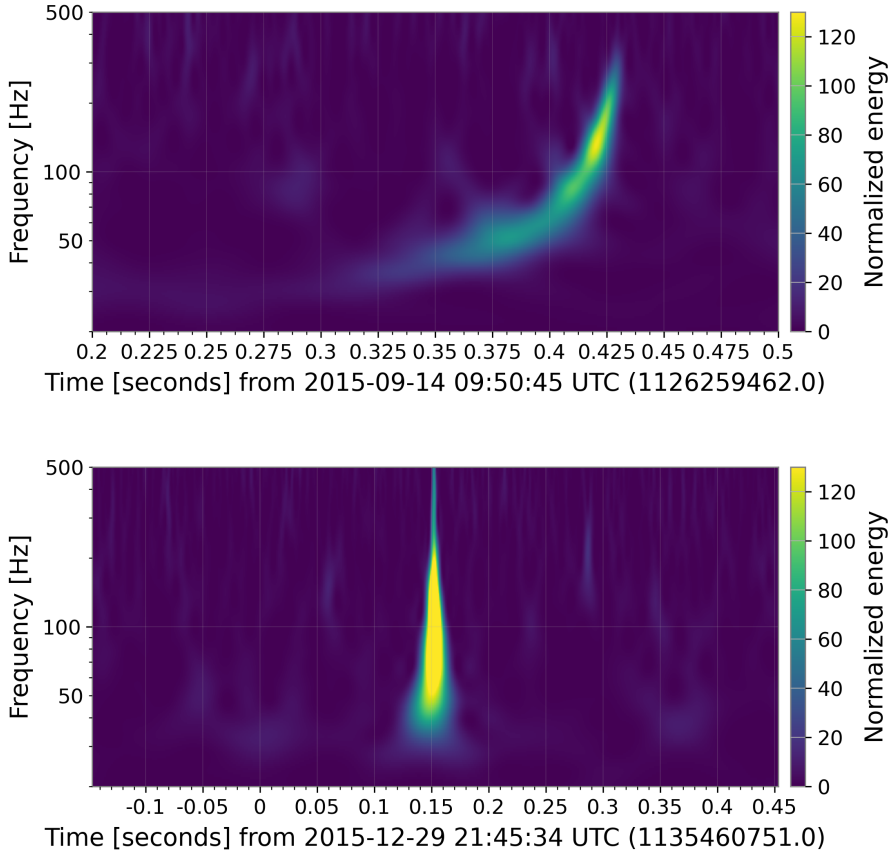


Figure 2.3: Q-transform representations obtained with GWPy [2] from LIGO Hanford detector data. The top panel shows the first detected GW signal, GW150914 [8], displaying the characteristic chirp pattern with increasing frequency as the BBHs merge. The bottom panel shows a typical glitch, illustrating a short non-Gaussian transient unrelated to any astrophysical. The GPS time of the glitch was selected from the Gravity Spy catalog [4].

where T denotes the duration of the segment of time and f_s is the sampling frequency.

Following the description in [25], and assuming a linear detector response, the recorded data can be represented as the sum of the GW signal and the instrumental noise

$$d = s(\theta) + n,$$

where $s(\theta)$ represents the deterministic signal, as a function of the source parameters θ, and n represents the stochastic noise arising from various sources, as discussed in 1.3. Here, the noise is assumed to be stationary and Gaussian.

In most of the cases, the noise is much larger than the signal, making the detection problem like searching for a needle in a haystack. The goal is to find a waveform template such that the residual between the strain data and the template contains mostly noise. This is the starting point for matched-filtering techniques.

Without going into the details of the method, an intuitive illustration of the filtering process can be presented as follows. Recalling that $d(t)$ represents the strain time series, $s(\theta)$ can be expressed as $h(t)$, the template assumed to represent the expected signal. Multiplying $d(t)$ by $h(t)$, integrating over the observation time window, and dividing by its duration, for large T we can write:

$$\frac{1}{T}\int_0^T dt\, d(t)h(t) = \underbrace{\frac{1}{T}\int_0^T dt\, h^2(t)}_{\text{positive definite} \sim O(1)} + \underbrace{\frac{1}{T}\int_0^T dt\, n(t)\, h(t)}_{\text{uncorrelated} \sim O(T^{-1/2})} \to h_0^2$$

So in the limit of large T the second term on the right vanishes and so the noise contribution is "filtered out". What remains is the average power of the waveform, proportional to h_0^2, which characterizes the strength of the oscillating function $h(t)$.

In more detail, in matched filtering, the waveform $h(t)$ is whitened and used to construct a filter, and the goal is to find the one that maximizes the signal-to-noise ratio (SNR). The optimal SNR is given by

$$(\text{SNR})^2_{\text{opt}} = 4\int_0^\infty \frac{|\tilde{h}(f)|^2}{S_n(f)}\, df \tag{2.1}$$

where $S_n(f)$ is the PSD of the noise and $\tilde{h}(f)$ is the Fourier transform of the signal $h(t)$.

In practice, the template is progressively shifted in time over the data, and the SNR is computed at each step. The optimal value corresponds to the time shift where the template best aligns with the signal. As an example, figure 2.4 shows the result of the matched filtering

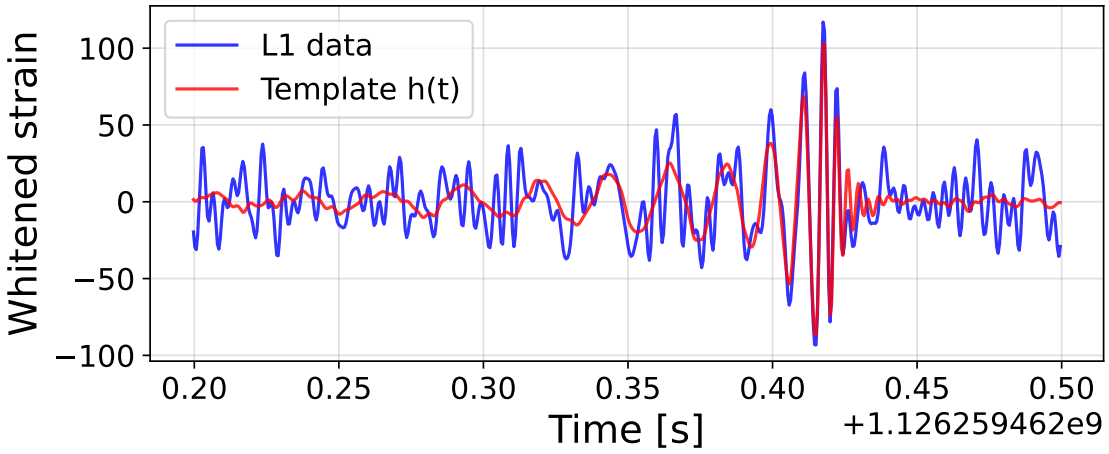


Figure 2.4: Result of the matched filtering pipeline applied to the LIGO Livingston detector data for GW150914 using the PyCBC package [6]. The whitened data are shown together with the aligned whitened template. The template corresponds to the h_+ polarization generated using the known signal parameters of GW150914 [9]. A $30-300$ Hz bandpass filter is applied to both the LIGO Livingston detector data and the template to suppress low- and high-frequency noise.

pipeline applied to the LIGO Livingston detector data for GW150914.

Since the true source parameters, and therefore the exact template, are unknown, the basic idea of matched filtering is to construct a bank of template waveforms, each acting as a filter, and to apply them with different possible starting times in order to identify the configuration with the best SNR.

This procedure is computationally intensive, as the SNR must be evaluated for a bank of templates with different source parameters and over many time shifts to identify the optimal match. Moreover, matched filtering is optimal under the assumption that the detector noise is stationary and Gaussian. But in practice, as already mentioned, GW data are affected by intermittent non-Gaussian transients, which can raise false alarms and make the detection of true signals more challenging. Coincidence tests between detectors play a crucial role in finding new signals, since instrumental noises are local in origin and can therefore be discarded.

Matched filtering is designed to identify whether a GW signal is present, rather than to provide an accurate estimation of the source parameters. For parameter estimation, offline methods are employed instead, typically based on Bayesian models [25].

In general, LVK analyses can be distinguished into two broad classes: online and offline. The first class provides a fast initial indication of the existence of a candidate GW event in the data and is used to send alerts to astronomers, who can look for the same event through other messengers, especially electromagnetic waves across all wavelengths. The offline searches are

performed later and allow researchers to confirm the presence of the GW event with greater confidence and to determine its parameters with higher accuracy.

The matched filtering technique is used in online analyses when the detectors are observing. Using a bank of template waveforms, the method searches for a match with the acquired data. This operation must be very fast in order to optimize detection and promptly alert the presence of a GW signal.

While traditional pipelines such as matched filtering are highly effective, their computational latency may become a limiting factor in the future due to the expected increase in the rate of GW detections, especially in the case of online searches. Consequently, alternative strategies and techniques are needed to identify new signals and efficiently reject noise transients.

3

Machine learning methods for GWs

In recent years, artificial intelligence has experienced extraordinary growth, establishing itself as a fundamental tool across many practical applications and scientific research. In particular, machine learning (ML) provides a framework that enables computers to "learn" directly from data and make predictions about novel data, thereby reducing the computational cost associated with explicit physics-driven simulations [34].

This chapter presents a theoretical overview of the ML concept, focusing on methods for classification and regression, which have potential applications for gravitational-wave (GW) data analysis. The theoretical framework presented follows the notation and definitions introduced in [34, 35].

3.1 Introduction to machine learning

As mentioned above, machine learning (ML) allows computers to learn directly from data, effectively converting "experience" into "knowledge". As described in [35], the input to an ML model consists of training data. These data may be associated with corresponding correct predictions, or not, leading to the distinction between supervised and unsupervised learning approaches. In the first case, the training data are associated with labels, which represent the correct output values or classes. The ML model is therefore supervised, meaning that it is guided by these labels to extract patterns and relationships from the data. Once trained, the model can make predictions on unseen data and assign the corresponding labels. In contrast, in unsuper-

vised learning, there is no distinction between training and unseen data, since the data are not associated with any labels. The goal of these methods is to identify hidden structures or group the data into categories.

In this thesis, only supervised learning methods are considered, and the following sections will focus on algorithms of this type.

3.1.1 Supervised learning framework

A supervised method operates on a domain set $\mathcal{X}$, which represents the set of all objects for which we want to make predictions. Each element in $\mathcal{X}$ is associated with a label from the label set $\mathcal{Y}$. The training set $\mathcal{S}$ is a finite sequence of pairs $(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathcal{X} \times \mathcal{Y}$, sampled from the domain with the corresponding labels. This set of pairs is used to train the ML model. A representation of the supervised learning method is shown in Figure 3.1. A training set is used to learn a model, for example composed of multiple layers, which connects the domain set to the predicted labels.

Formally, the goal of an ML algorithm is to find a prediction rule $\boldsymbol{h} : \mathcal{X} \to \mathcal{Y}$ that best approximates the true function, unknown by the algorithm. Finally, the prediction rule is evaluated using a loss function, which quantifies the success of the algorithm. This is the probability that the predicted label does not match the correct label.

To minimize the loss, optimization algorithms are used to efficiently compute the gradient. Examples include stochastic gradient descent (SGD) and Adam [36, 37], which are widely used and help to accelerate convergence.

Some of the most common supervised learning problems, which share the same domain set $\mathcal{X}$ and training data, are:

- Binary classification, where the label set is $\mathcal{Y} = \{0, 1\}$.
- Multiclass classification, where the label set is $\mathcal{Y} = \{0, 1, 2, \ldots, \boldsymbol{K} - 1\}$, with $\boldsymbol{K}$ being the number of classes.
- Regression, where $\mathcal{Y} \subseteq \mathbb{R}$, and so the output space is continuous.

Different types of loss functions are adopted in ML depending on the learning problem.

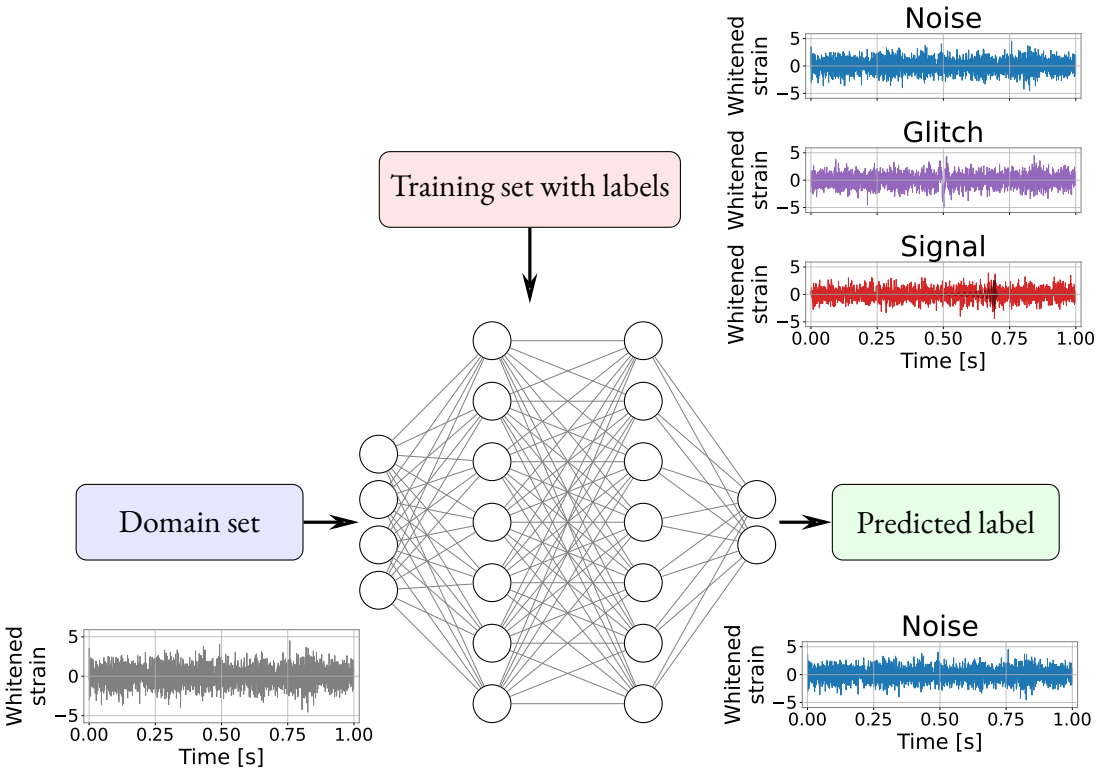


Figure 3.1: Schematic representation of supervised learning. A domain set is fed into a ML model. Before making predictions, the ML model is trained using a labeled training set. After training, the model can make predictions on the domain set. The sketch at the center of the figure represents a well-known ML model called a neural network (NN), which will be described later.

In classification tasks, the simplest option is the 0–1 loss, which only checks whether the predicted label $\boldsymbol{h}(\boldsymbol{x})$ coincides with the true one $\boldsymbol{y}$:

$$\ell_{0-1}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 0, & \text{if } \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{y}, \\ 1, & \text{otherwise.} \end{cases}$$

A more informative alternative for classification is the cross-entropy loss. Here, the label vector $\boldsymbol{y}$ is one-hot encoded, i.e. $\boldsymbol{y_i} = 1$ for the true class and otherwise 0. This loss measures how well the predicted probability distribution $\boldsymbol{f_i}(\boldsymbol{x})$ matches the true class $\boldsymbol{y_i}$ [38].

For regression problems, two of the most common choices are the squared loss ($\boldsymbol{L}_2$) and the absolute loss ($\boldsymbol{L}_1$), defined as:

$$\ell_{\text{sq}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{y})^2, \quad \ell_{\text{abs}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{y}|.$$

The $\boldsymbol{L}_2$ gives higher weight to large deviations, whereas the $\boldsymbol{L}_1$ loss tends to be less sensitive to outliers.

3.2 Model selection and validation

In ML, there exist different algorithms and models to solve specific types of problems. Every algorithm can be associated with a hypothesis class, which represents the set of all possible prediction functions that the model can learn from data.

For example, in polynomial regression, the hypothesis class corresponds to the set of all polynomials up to a certain degree (see figure 3.2). The degree acts as a hyperparameter that controls the model's flexibility: higher degrees allow the model to better fit the training data, however, excessive complexity may lead to overfitting, as the model fails to generalize to new, unseen data.

The goal of model selection is to identify the best trade-off between model complexity and generalization ability.

As previously described, the dataset is initially divided into a domain set, which contains all possible data for making predictions, and a training set, consisting of data that are known and labeled. In practice, the training set is split into three groups: training data, validation data, and test data.

The training data are used to learn a prediction function $\boldsymbol{h}$ within the chosen hypothesis class.

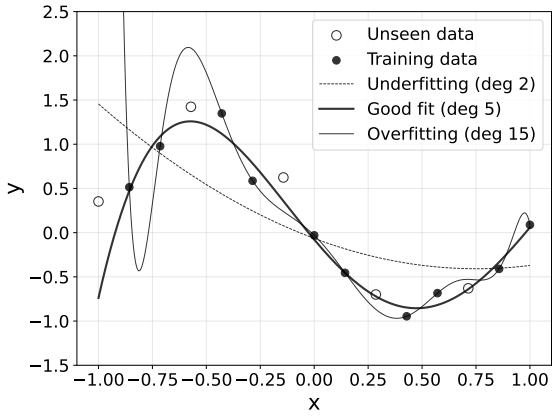


Figure 3.2: Example of polynomial regression fits for different degrees. When the degree is too low or too high, the model goes to underfitting or overfitting, as the unseen data (not used for the fit) are not well matched. This figure is generated using least squares fitting methods and it is simple example to illustrate the basic concepts of model complexity.

During training, the algorithm minimizes the loss on the training data; however, this does not guarantee good generalization to unseen data. The test data are used to evaluate whether the trained model generalizes well, providing, as a final step, an estimate of the model's true error. The test set should be used only once to ensure that the prediction function does not indirectly adapt to it. To overcome this limitation, validation data are used during the training procedure to select the best hypothesis $\boldsymbol{h}$ among those learned from the training data. The validation data are not used to train the model, but only to assess the performance of different hypotheses. Therefore, the loss computed on the validation set is generally more stable with respect to the loss computed on the training data, and it helps the algorithm avoid overfitting.

Regularization

The goal of an ML model is to minimize the loss function. However, as mentioned above, in many cases, a model with high complexity may lead to overfitting. To mitigate this issue, a regularization term is added to the loss function. This additional term acts as a stabilizer. As a result, complex models are less favored. Different forms of regularization can be applied depending on the specific task.

3.3 Machine learning models

As discussed above, a hypothesis class refers to the set of all possible prediction functions. In practice, it is determined by the specific ML algorithm and its hyperparameters, which define

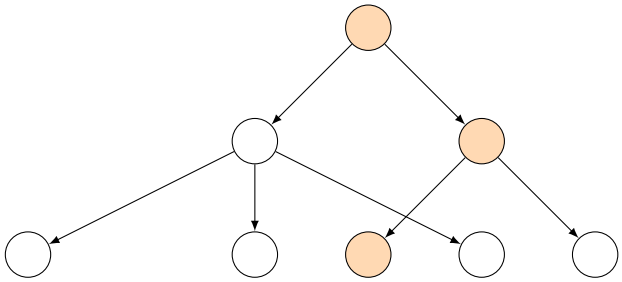


Figure 3.3: Example of a decision tree. The orange path shows the steps to reach the leaf.

the space of possible models. During training, the algorithm learns the optimal set of parameters that identifies the best prediction rule within this class. For example, in a linear model the parameters are a set of values θ, such that the prediction for the i-th sample is given by $y_i = \sum_j \theta_j x_{ij}$ [39].

With this theoretical background, the next section introduces the ML models used in this thesis, based on decision trees and neural networks.

3.3.1 Decision trees and XGBoost

A decision tree is a predictive model structured as a set of tree paths. Starting from the root, at each fork the model "asks a question" based on the ***features*** of the object to be classified and decides which branch to follow until it reaches a leaf, as shown in figure 3.3. The leaf of the tree corresponds to the final prediction.

Decision trees do not simply classify data: by following a path from the root to the leaves, each example can be assigned a specific score. This enables the trees to address both classification and regression tasks, providing more detailed information than a simple class label [39].

The limitation of a single tree is that small changes in the training data can lead to very different tree structures, making the algorithm highly sensitive to the data used for training.

A tree ensemble model extends the concept of a single decision tree and combines multiple decision trees to produce more robust predictions. An example is the random forest, where several decision trees are trained independently on random subsets of the training data. The same idea applies to the features used for training. In a single decision tree, all features are considered. In a random forest, however, each tree is trained on a different subset of features. Finally, the individual tree predictions are averaged or combined by majority voting to produce the final prediction, which is more stable and less sensitive to variations in the training data and in the selected features.

Another type of ensemble model, which, like random forest, combines the predictions of multiple decision trees, is boosting [39]. The main difference with random forest lies in the training process. In random forest, as mentioned above, the trees are trained independently of one another. In boosting, each new tree is trained by taking into account the errors made by the previously trained trees, allowing the model to progressively correct them and improve overall predictions [39].

This chain of decision trees is constructed by optimizing a loss function which, combined with a regularization term, penalizes overly complex trees and reduces the risk of overfitting. This approach is the basis for algorithms such as eXtreme Gradient Boosting (XGBoost) [39, 21].

To evaluate the quality of a tree structure, the algorithm uses the structure score, which measures how much a split improves the loss function while also considering the regularization term. This score guides the algorithm in selecting splits, helping to construct simpler trees and reduce the risk of overfitting [39].

Thanks to these characteristics, XGBoost is often more robust than individual decision trees or traditional ensemble models.

3.3.2 Neural networks

In the previous section, XGBoost was presented as a model that evolves and improves at each iteration. Another algorithm that follows a similar iterative learning process, though very different in practice, is the neural network (NN).

Neural networks are inspired by the structure of the human brain, where fundamental units, called neurons, process information through connections. In an artificial neural network the connections between the neurons are represented by numerical weights. These weights represent the parameters that the ML algorithm learns during training. As shown in figure 3.1, these neurons are organized in layers, and information flows through these connections to identify patterns and perform predictions.

Each neuron receives as input the weighted sum of the outputs from the neurons in the previous layer. The output of a neuron is then obtained by applying a nonlinear function to this weighted sum. These functions, called activation functions, accelerate the training process, and help the network to minimize the loss.

Since the loss can only be computed at the output of the last layer, after the initial forward propagation a backpropagation is performed, updating the weights in the direction opposite to

the gradient of the loss, adjusting the parameters to minimize the error. This process of forward and backward propagation is called epoch. The training algorithm can be run for a specified number of epochs, which is chosen according to the problem and dataset.

Since these layers can become extremely dense, a common method to prevent overfitting is dropout. This involves randomly "turning off" some neurons with a certain probability during training. At each step, the network is trained only on a subset of neurons, ensuring that no single neuron becomes too influential. This stabilizes training and reduces the risk of overfitting [34].

These networks can be fully connected, meaning that each neuron is linked to all neurons in the previous layer, as in figure 3.1. This results in a large number of weights and, therefore, many parameters to learn, some of which may be unnecessary. Data such as images or time series often exhibit predictable local patterns. Convolutional neural networks (CNNs) exploit these local correlations, reducing the number of parameters that need to be learned [34].

Convolutional neural networks

CNNs were originally introduced for image classification, where they operate on two-dimensional data and are therefore referred to as 2D−CNNs [34]. The same principles can, however, be adapted to one-dimensional signals, giving rise to 1D−CNNs.

As mentioned above, in CNNs, the fully connected layers of classical NNs are replaced by convolutional layers. These layers consist of multiple filters; a set of parameters, which allows the network to detect local patterns. The filter slides over the input data, producing outputs that form a feature map. The size of a filter is called the kernel size, while the stride defines the step size of the filter as it moves. This process generates a feature map for each filter, which constitutes the input of the next convolutional layer. Feature maps are often downsampled using pooling layers to reduce their resolution. Finally, all feature maps from the last convolutional layer are flattened and passed to a fully connected layer, which outputs the final predictions.

When applying a convolution with a filter, the resulting feature map tends to shrink. This occurs because the kernel cannot extend beyond the boundaries of the input. Therefore, the information at the edges is discarded, producing a feature map shorter than the original input.

To address this issue, zero padding is introduced, meaning that zeros are added to the borders of the input. This allows the filter to operate the convolution also at the extremes of the input data [34].

InceptionTime

InceptionTime (IT) is a machine learning model specifically designed for time series classification [40]. It consists of an ensemble of deep convolutional neural networks (CNNs) built to handle the characteristics of time series, which differ from images because their information is distributed over time. Each classifier (module) applies multiple filters of varying lengths simultaneously to the input time series. This allows the network to automatically extract relevant features from both long and short sequences, improving its ability to classify the data. Each Inception module first summarizes the input multivariate time series into a one-dimensional representation called the "bottleneck" layer. Then, multiple filters of different sizes are applied to generate several feature maps, capturing information at different resolutions. Finally, the dimensionality is reduced with max pooling before passing the output to the next module [20]. In this way, the architecture is able to detect both local and global structures in the signal, improving its classification performance.

Based on the considerations discussed in this chapter, NNs and XGBoost are ML methods especially effective for GW time-series analysis.

4

Characterization and generation of non-Gaussian noise

The detection of GWs is complicated by the presence of non-Gaussian noise, known as glitches. These instrumental artifacts exhibit a wide range of morphologies, frequencies, and durations. Consequently, it becomes essential to classify and analyze glitches. Thanks to studies of this kind, several glitch sources have been identified and their occurrence has been reduced. However, improvements in detector technologies and increased sensitivity also lead to the appearance of new glitches, generated by environmental or instrumental noise that was previously too weak to be detected. For example, at the LIGO Livingston detector the rate of glitches recorded in O3a and O3b was about 5 times higher than O2 [1].

This chapter addresses the problem of characterizing and generating non-Gaussian noise. It begins with an analysis of the properties of real glitches using public catalogs such as Gravity Spy [4], which provide labeled examples from many different classes of these artifacts. Then, the chapter focuses on simulating a particular family of glitches, whose origin may be related to Barkhausen noise [10]. This study allows the construction of a controlled dataset for training algorithms capable of distinguishing noise, glitches, and astrophysical signals.

4.1 Characterization of glitches

In LVK strain data, glitches appear as short bursts of excess power that cannot be linked to known astrophysical sources. They typically last only a few seconds and can have different morphologies [3]. Glitches are detected using the Omicron algorithm, a software tool that analyzes time-frequency maps of GW data [33]. These maps are constructed in a manner similar to the Q-transform, producing spectrograms of whitened data.

For some glitches, the environmental or instrumental origin is known, while for others the source remains unknown [3]. Studying and classifying glitches is therefore essential for identifying noise sources and improving data quality.

4.1.1 Gravity Spy classification model

The Gravity Spy project aims to morphologically classify glitches using machine learning (ML) techniques [3]. In particular, Gravity Spy uses convolutional neural networks (CNNs) trained on spectrograms generated with Omicron to classify the different glitch families. By examining data from the O1, O2, and O3 observing runs, Gravity Spy has accumulated millions of classified glitches with SNR > 7.5. For O3 these glitches are divided in 23 classes (previously shown in figure 1.3 in chapter 1).

One result from Gravity Spy is that some glitch classes occur much more frequently than others. For example, during O3 at the LIGO Livingston detector, the most common classes of glitch were scattered and fast scattered light, which together accounted for about 50% of all detected glitches [3].

Another often problematic class is that of blips, which sometimes exhibit a morphology similar to that of a true GW signal. These glitches can be grouped into a broader family of transient events, which exhibit similar characteristics and are labeled in Gravity Spy [3] as follows:

- Blips: extremely short glitches lasting about $\sim$ 0.04 s, with power spread over a wide frequency range. They can affect searches for high-mass BBH systems. Blips can occur repeatedly over short time intervals and are classified as repeating blips.

- Low-frequency blips: similar to blips but with most of their power concentrated at lower frequencies, typically between 10 and 50 Hz.

- Extremely loud: glitches characterized by broad frequency content and high SNR, often causing spectrogram saturation.

- Tomte: very short glitches at relatively low frequencies, with a characteristic triangular shape.

- No glitch: a category in which no clear excess power is visible in the spectrogram.

- Koi fish: glitches with high SNR and a wide frequency range. They resemble blips but are often confused with extremely loud. They exhibit the characteristic low-frequency "pectoral fins".

It is reasonable to think that these kinds of glitches might originate from the same physical process. This process should produce short-lived impulses whose amplitudes can vary substantially at different scales.

The Q-transforms of a sample of these particular classes of glitches, selected from the Gravity Spy catalog [4], are shown in figure 4.1.

4.1.2 Possible physical origins of glitches

Although the morphological classification provided by Gravity Spy offers a detailed overview of the different glitch families, the physical origin of many of these transient events remains unclear, while others show clearer correlations with specific environmental or instrumental conditions. For instance, the occurrence of scattered and fast scattering glitches is strongly correlated with ground motions due to human activity or earthquakes [41]. For the family considered in the previous section, as already mentioned, the common morphology suggests a shared physical mechanism, and one possible explanation involves electromagnetic effects inside the interferometers [10]. The presence of ferromagnetic materials in LVK interferometer components, such as circuits, coils, and magnets can give rise to a particular noise, known as Barkhausen noise. As explained in the following section, this source of noise produces sudden jumps in the magnetization of ferromagnetic materials, which can manifest as impulses on strain data.

4.2 Barkhausen noise

To understand its origin, it is first necessary to review the magnetic properties of materials. This section summarizes the main concepts discussed in [10].

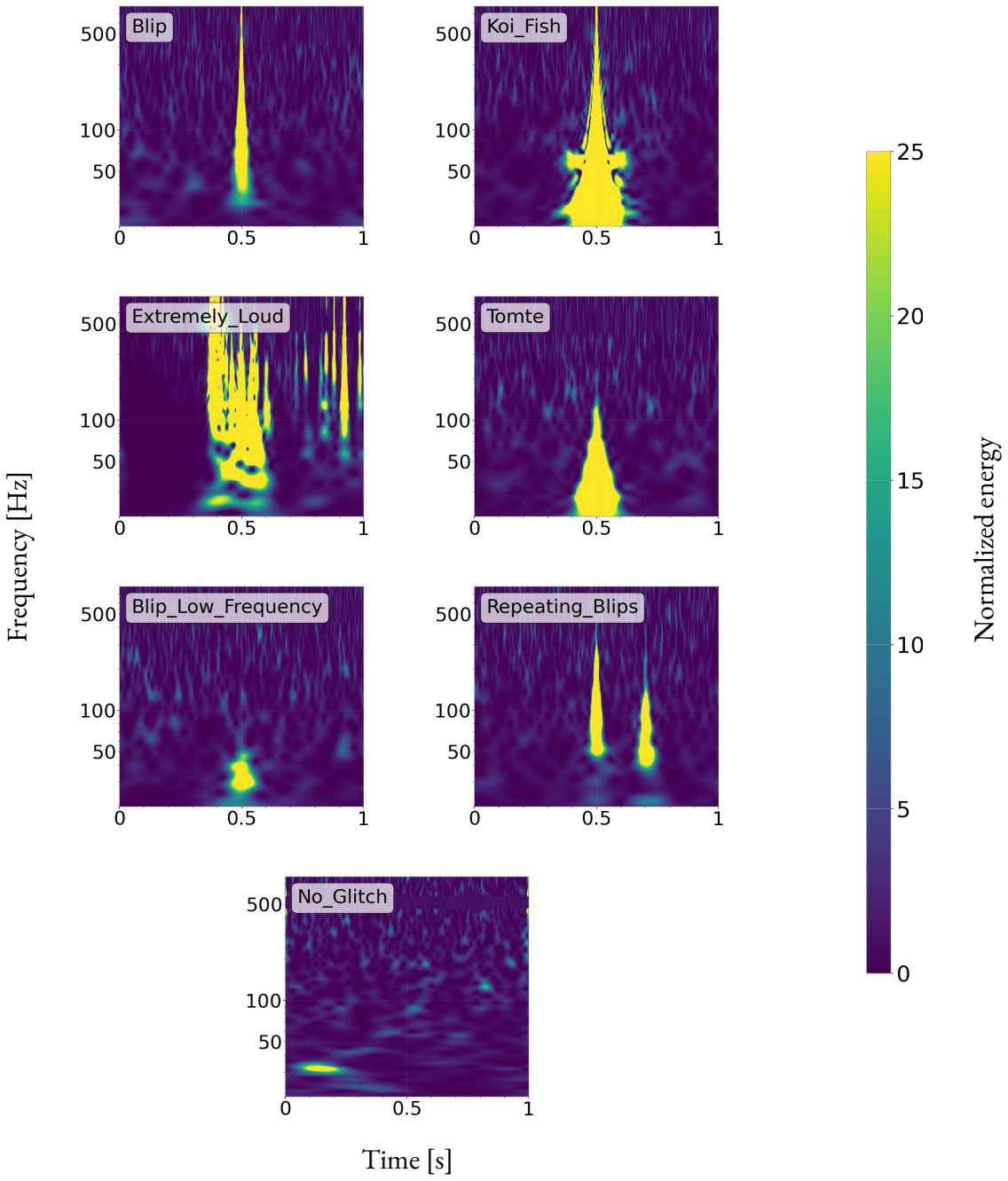


Figure 4.1: Q-transform representation of a set of glitches classified by Gravity Spy. These glitches come from GWOSC time-series data recorded by the LIGO Livingston detector during the O3b run. The labels are those from Gravity Spy, selected with a confidence level greater than 90% [4]. These time-frequency maps were performed using the `q_transform()` function from the GWPy package [2], with the choice of the Q-range between 8 and 64 and normalized energy between 0 and 25.

4.2.1 Introduction to magnetism in materials

Materials can interact with magnetic fields and are classified into different categories. Diamagnetic materials are weakly affected by a magnetic field, ferromagnetic materials respond strongly, and paramagnetic materials exhibit a weak attraction toward the field, behaving in between diamagnetic and ferromagnetic materials. These properties depend on the microscopic characteristics of materials, in particular on the coupling of the electron spins within the atoms that make up the material.

The motion of an electron around an atom's nucleus can be regarded as an electric current, which generates a magnetic field and, consequently, a magnetic moment associated with the electron. In an atom, the magnetic moments of electrons cancel out if all electrons are paired. When unpaired electrons are present, the total magnetic moment does not vanish, and the sum of these moments gives rise to the magnetization field. This field is only significant over short distances. Consequently, under conditions of strong magnetization, the magnetic moments of the electrons in each atom are directly influenced by those of neighboring atoms.

In the presence of an external magnetic field, the energy of an atom depends both on the intensity of the external field and on the field generated by the unpaired electrons of neighboring atoms. In the case of strong magnetization, as in ferromagnetic materials, the magnetic dipoles tend to align with those of their neighbors to reach a minimum energy, forming regions of aligned dipoles called magnetic domains. As a result, each domain exhibits strong local magnetization, while the average magnetization of the entire material may be close to zero if the domains are oriented in different directions.

In a crystalline structure, certain directions are preferred for the alignment of magnetic dipoles, while others are less favorable. This difference creates anisotropies in the magnetization field of the material. These preferred directions are known as easy axes, and a crystal structure may contain more than one. In ferromagnetic materials, the magnetization within each domain spontaneously aligns along one of these easy axes, as this configuration minimizes the system's energy.

There is no abrupt transition in magnetization between one domain and another. Instead, the change occurs through intermediate regions called domain walls. Within these walls, the magnetization rotates gradually, as shown in figure 4.2; in particular, the angle between neighboring atoms decreases as the number of atoms in the wall increases.

The energy of a domain wall depends on both the exchange interaction between adjacent atoms and the material's magnetic anisotropy. Consequently, under minimum energy condi-

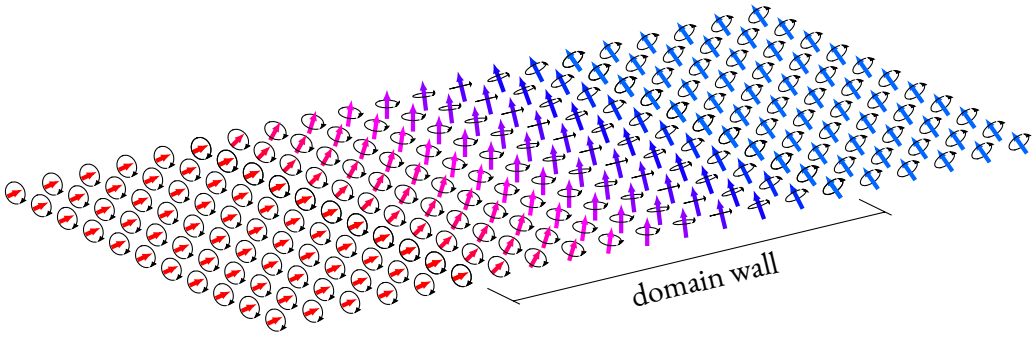


Figure 4.2: The figure shows a representation of two magnetic domains connected by a domain wall. As can be seen, the orientation gradually changes from one magnetic domain to the other. The figure is adapted from [10].

tions, stronger exchange interactions and weaker anisotropies lead to thicker domain walls.

As mentioned above, when an external magnetic field is applied, atomic dipoles tend to align with it. Consequently, configurations that were previously stable may become unstable, leading to new equilibrium states.

4.2.2 Magnetic wall dynamics

When an external magnetic field is applied, dipole alignment can occur in two ways. In the first case, all misaligned dipoles would simultaneously orient along the external field, but this requires too much energy and is therefore unlikely. It is more probable that the less aligned domains shrink in size, causing the motion of the domain walls. In this way, the magnetization field never reaches full saturation and maintains a memory of the dipoles' past alignment.

When domain walls move under the influence of an external magnetic field, they can encounter lattice imperfections, impurities, or mechanical stresses. As a result, the magnetic dipoles face resistance to alignment, slowing down the wall motion. Consequently, the number of dipoles actually aligned is lower than the ideal number, leading to a deviation from the expected magnetization curve, which represents how the material's magnetization varies with the applied field. If the applied field increases sufficiently, it can overcome defects in the domain walls and align the trapped magnetic dipoles. This results in an abrupt change in the material's magnetization, known as a Barkhausen jump. This is a stochastic process that depends on the microscopic configuration of the domains and the imperfections present in the material. The combination of these jumps produces noise in the magnetization curve, referred to as Barkhausen noise, as illustrated in figure 4.3.

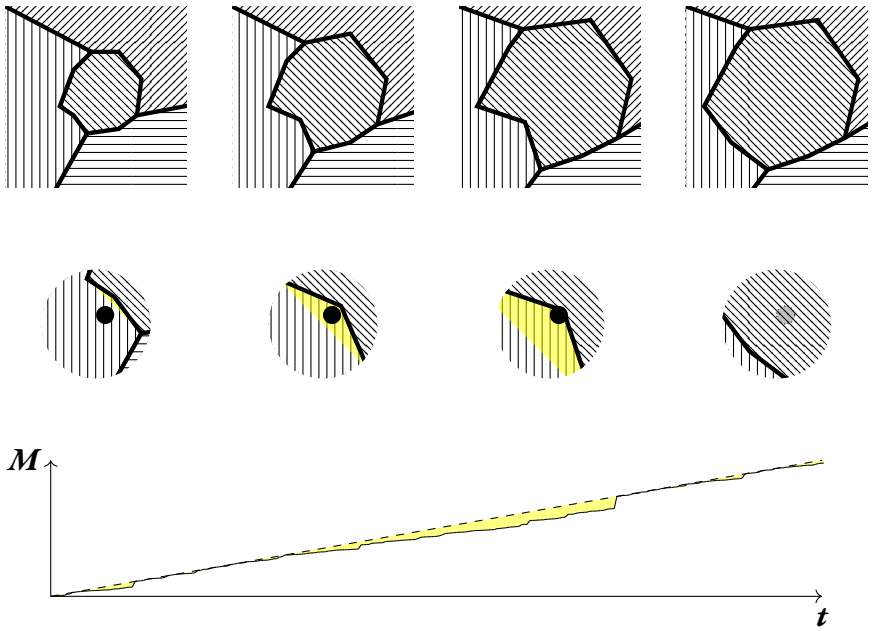


Figure 4.3: Top: The figure illustrates the effect of an increasing magnetic field on the domain walls. It shows how the domain walls move over time, aligning the magnetic dipoles in a specific direction. Middle: When the domain wall encounter a defect, represented as a black circle, it gets pinned at the imperfection. The region that would normally be occupied by the domain wall is colored in yellow. This pinning is eventually overcome as the magnetic field continues to increase. Bottom: Magnetization as a function of time, with the dashed line indicating the expected behavior. The figure is adapted from [10].

4.3 A model for Barkhausen noise

As seen above, Barkhausen jumps manifest as discrete jumps in the magnetization curve. These jumps can be small, caused by small imperfections that are more likely to occur, or larger, due to more extensive and rare defects.

Barkhausen noise is therefore a process in which both large and small jumps derive from the same physical mechanism and behave similarly across different scales, a property known as "universality". Phenomena of this type are described by crackling noise, typically used to represent events such as avalanches and earthquakes [42].

Following [43], let us consider a signal $x(t)$ and a fluctuation of the signal, for example generated by a Barkhausen jump. This signal can be modeled as an increase of amplitude from a value $x(0)$ and returns to the same value after a time T, which is defined as the duration of the pulse. In particular, in this interval one can think of $x(t)$ as the trajectory of a "walker" $x(t+\delta t) = x(t) + \xi(t)$, where $\xi(t)$ is an added random component, subject to the constraint $x(0) = x(T)$.

The average of several events, $\langle x(t)\rangle_T$, each of duration T, characterizes the typical pulse shape. The average behavior of these stochastic processes scaled as:

$$\langle x(t)\rangle_T = T^{\mu} f\Big(\frac{t}{T}\Big), \tag{4.1}$$

where $f(t/T)$ is a universal scaled function and μ is a characteristic exponent of the power-law [43]. The function $f(t/T)$ is universal in the sense that it captures the shape of the fluctuation independently of its duration. In case of uncorrelated noise the universal law is proportional to a semicircle.

To build the walker trajectory for Barkhausen noise, different distributions can be used. A distribution that naturally fits the characteristics of this source of noise is the Levy flights. This distribution is defined as

$$P(\xi) \propto \frac{1}{\xi^{\alpha+1}}$$

Unlike a Gaussian, it has heavy tails, which, although rare, allow for large jumps. This effect happens when the power-law index α is in the range $]0, 2[$, where the variance of the distribution is infinite. Numerically it can be shown that Levy flight obeys equation 4.1 for $\mu = 1/\alpha$ [43]. For this reason, a Levy flight walk can model a crackling pulse and can therefore be used to simulate Barkhausen noise.

4.4 Glitch generation pipeline

The previous section introduced a model inspired by crackling-noise phenomena, which can be used to simulate a signal generated by Barkhausen jumps. In this work, stochastic processes of this kind, characterized by short and irregular bursts, are assumed to give rise to some of the glitch classes observed in detector strain data.

This section describes the generation of synthetic pulses based on the presented model and their injection into real real LVK data.

4.4.1 Data preparation

The simulated glitches are injected into strain data obtained from GWOSC. The choice to use real data instead of a simulated noise background is motivated by the goal of comparing the simulated glitches with the real ones present in the real data. This approach ensures that the analysis is carried out under the same conditions as the real glitches, making the comparison more realistic.

For this purpose, this work uses data from the LIGO Livingston detector during the O3b run, that pass the category 3 (CAT3) of data quality requirements, sampled at 16 kHz. The procedure involves downloading several data segments which are then resampled to 2048 Hz using the `resample()` method of the GWPy package [2]. This resampling reduces the data volume while remaining sufficient for the typical frequencies of glitches. With a sampling rate of 2 kHz, the Nyquist frequency is 1 kHz, which adequately captures both blip glitches and typical GW signals.For each data segment, the ASD is computed using the `asd()` method from the GWPy package [2], and the segment is subsequently whitened. The ASDs calculated for each segment are then used to whiten the simulated glitches before injection, ensuring that both the noise and the glitches are represented under the same conditions. A high-pass filter of 20 Hz is applied to each segment to remove low-frequency noise. In addition, the first and last second of each segment are discarded, as edge regions may be affected by artifacts introduced by both the whitening and the high-pass filtering steps. Each segment is divided into 1-second intervals in order to build a set of time series for synthetic glitch injection. Finally, all segments identified as containing glitches in the Gravity Spy catalog were removed, yielding a dataset largely free of artifacts. However, as explained previously, Gravity Spy only cataloged glitches with SNR greater than 7.5; therefore, segments containing lower-SNR glitches may still be present. Since the LVK data contain candidate GW signals, these have been removed.

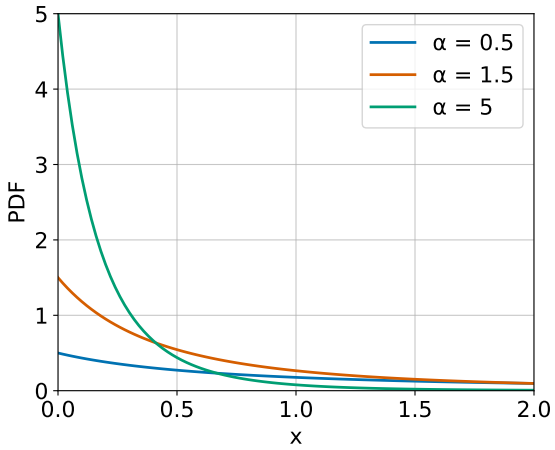


Figure 4.4: Probability density function (PDF) of the Pareto distribution for three different values of α. Higher α values concentrate the distribution at smaller x, while lower α values result in a flatter distribution, extending to larger x.

4.4.2 Pulse generation

To generate the pulses injected into the strain data, each step of the random walk is drawn from `random.Generator.pareto()`, a function from NumPy [44], which produces random variables following a power-law distribution. The PDF used is shown in figure 4.4 for three values of power-law index. The figure illustrates how smaller values of α produce heavier tails, making the occurrence of very large jumps non-negligible. For larger values of α, the distribution becomes more peaked.

The impulse is constrained: its value starts at zero, rises, and then returns to zero. To generate it, each step is multiplied by a randomly chosen positive or negative sign, creating a random walk until the signal changes sign. The last value is, therefore, set to zero. To generate impulses following a Levy flight, the power-law index is chosen within the interval $]0, 2[$. The random walk evolves for a number of steps not exceeding 2000; any trajectory that exceeds this maximum is discarded and regenerated. The choice of the maximum number of steps ensures that the generated pulse remains within the segment window of 1 second. It should be noted that the number of steps the impulse takes is variable and depends on the trajectory. This mainly depends on the distribution and the choice of the parameter α, which determines the heaviness of the distribution's tail. Consequently, for small values of α, walks with few large steps are more likely, whereas for large values of α, walks with many small steps are more probable. Once the trajectory is obtained, the signal is scaled by an amplitude factor A to bring it to a realistic physical scale. The parameter A does not uniquely determine the amplitude of the im-

pulse; it acts only as a multiplicative factor. The overall amplitude depends on the trajectory of the impulse multiplied by $\boldsymbol{A}$. Consequently, knowing the generation parameters $\boldsymbol{\alpha}$ and $\boldsymbol{A}$ is essential to generate realistic glitches.

4.4.3 Glitch injection into data

Once the impulse is generated with the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{A}$, it must be brought to the same conditions as the data into which it will be injected. This means that the impulse must be whitened using the same ASD applied to the specific data segment in which the impulse is injected, and then filtered with a high-pass filter at 20 Hz. Only after these steps can it be injected into the data, thus generating a glitch. An example of a glitch generated using the procedure described above is shown in figure 4.5. At the bottom of the figure, a time-frequency map of the glitch after the injection is also shown.

4.5 Analysis of the generated glitches

To understand the characteristics of the generated glitches, it is useful to define a set of ***features*** that describe the properties of the glitch population as a function of the generation parameters $\boldsymbol{\vartheta} = \{\boldsymbol{\alpha}, \log_{10} \boldsymbol{A}\}$. From now on, the parameter $\boldsymbol{A}$ is shown as $\log_{10} \boldsymbol{A}$, since it is a scale parameter and its main component is the exponent. The features used in this work are parameters based on those defined in [10], denoted collectively as the set $\boldsymbol{\lambda}$. These parameters characterize both the temporal and spectral properties of glitches.

Time-domain parameters:

- `max_amp`: maximum amplitude of the signal, corresponding to the central peak.
- `FWHM`: full width at half maximum of the central peak.
- `width`: width of the central peak.
- `der`: curvature of the central peak.
- `integ`: integral of the signal.
- `Pinteg`: integral of the central peak.

Frequency-domain parameters:

- `peak`: maximum amplitude of the spectral peak.

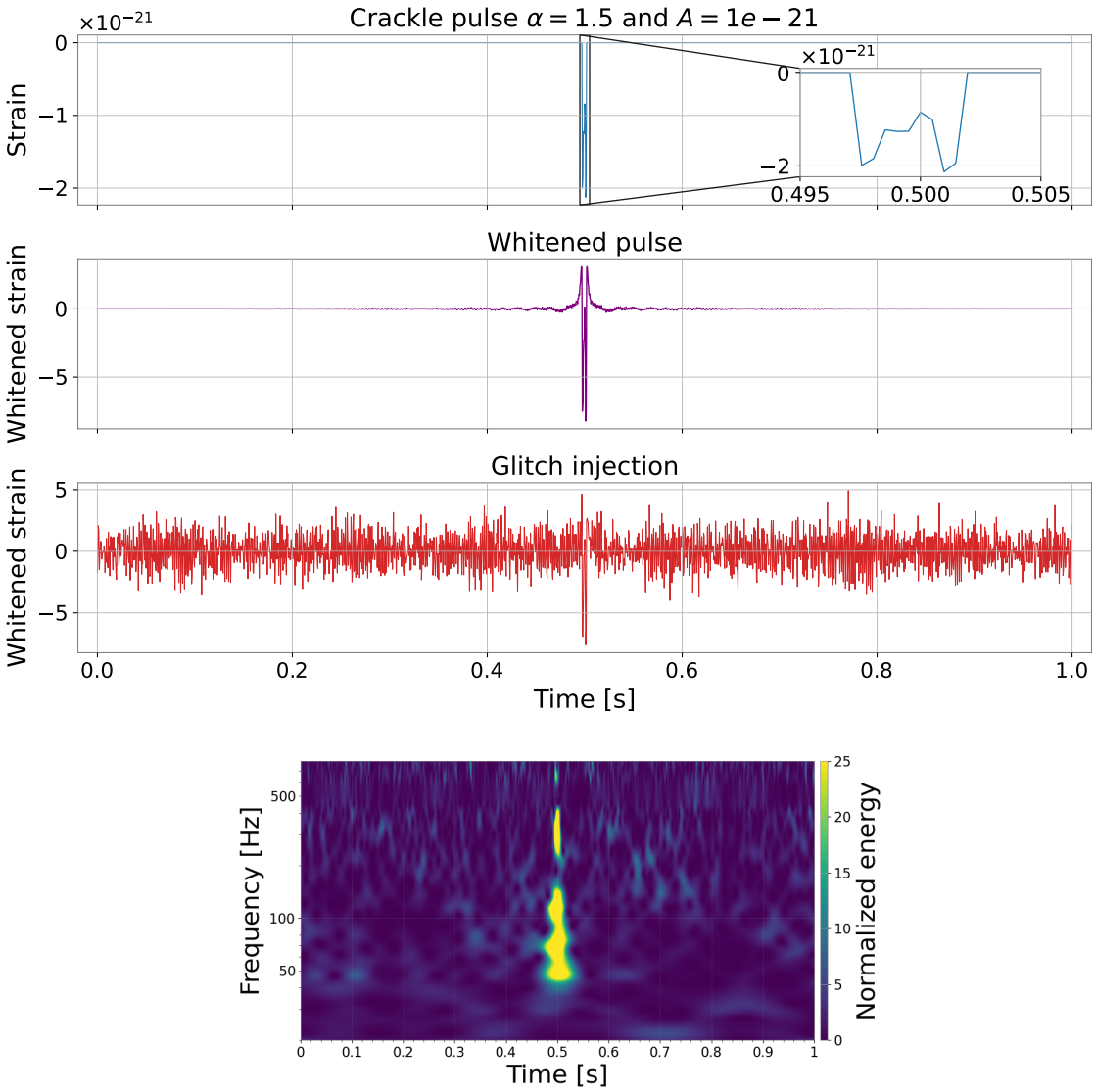


Figure 4.5: Top: Illustration of the three steps used to generate a synthetic glitch. First, an impulse is generated with parameters α and A. A zoomed-in view on the right clearly shows the impulse. The impulse is then whitened using the ASD of the noise corresponding to the segment where the injection will occur, and is then filtered with a high-pass filter at 20 Hz. Finally, the glitch is injected into the noise segment at GPS time 12566656219.5. Bottom: Q-transform of the resulting glitch.

- `max_freq`: frequency corresponding to the spectral peak.
- `bw`: full width at half maximum of the main spectral peak.
- `bw2`: dispersion of the spectrum around the main peak.

Without going into too much detail, the following explains how they are calculated. The first parameter, namely the `max_amp`, is obtained by computing the absolute maximum of the signal. The `fwhm` measures the width of the peak at half of its maximum amplitude, while the `width` is measured as the distance between the first zero crossings of the signal surrounding the peak. The `der` parameter is computed as the second derivative of the peak. It provides information about the peak's curvature, indicating whether it is sharp or more gradually varying. This is possible because whitening suppresses certain frequency components, particularly those dominated by noise, smoothing the signal and reducing the sharp jumps inherent to the random walk, which results in a continuous signal rather than jagged lines. The difference between `integ` and `Pinteg` is that the first one measures the integrated amplitude around the peak within a fixed time window, whereas `Pinteg` is the integral of only the portion of the signal confined between the zero-crossings that delimit the central peak.

All frequency-domain quantities are computed using the Fourier transform of the signal. `peak` and `max_freq` correspond to the maximum amplitude of the spectrum and its associated frequency. `bw` is calculated analogously to the time-domain `fwhm`, measuring the width of the spectral peak at half of its maximum amplitude. Finally, `bw2` represents the dispersion of the spectrum around the dominant peak, computed as the weighted standard deviation of the logarithm of the frequencies, weighted by the spectral amplitude:

$$\mathrm{bw2} = \sqrt{\frac{\sum_{i=1}^{N-1} |\boldsymbol{X}(\boldsymbol{f_i})| \left(\log(\boldsymbol{f_i}) - \bar{\boldsymbol{f}}_{\log}\right)^2}{(\boldsymbol{N}-2)\sum_{i=1}^{N-1} |\boldsymbol{X}(\boldsymbol{f_i})|}}, \quad \bar{\boldsymbol{f}}_{\log} = \frac{\sum_{i=1}^{N-1} \log(\boldsymbol{f_i})\,|\boldsymbol{X}(\boldsymbol{f_i})|}{(\boldsymbol{N}-1)\sum_{i=1}^{N-1} |\boldsymbol{X}(\boldsymbol{f_i})|}$$

Here, $\boldsymbol{X}(\boldsymbol{f_i})$ is the Fourier transform of the time series and $\boldsymbol{f_i}$ are the corresponding frequencies, with $\boldsymbol{i} = 1, \ldots, \boldsymbol{N}-1$ to exclude the zero-frequency bin.

4.5.1 Data selection and glitch generation

The analysis considers segments covering the period from November 1, 2019 (GPS time 1256665618) to November 6, 2019 (GPS time 1257109230). Only segments with a duration of at least 1000 seconds were selected. This choice of segments longer than 1000 s is somewhat arbitrary, but it ensures a reliable estimation of the ASD.

This selection yields a total of 272997 seconds, corresponding to approximately 3.16 days of data. Two GW events, GW191105_143521 and GW191103_012549 [1], were identified and removed. The same procedure was applied to the glitches identified by Gravity Spy [4]. A total of 7211 seconds containing glitches were identified, distributed across 23 classes, of which 2173 belong to the classes relevant for this work with an ML confidence level greater than 90% [4]. The total number of glitch segments accounts for both the peak position and the full duration of each glitch, as reported in Gravity Spy. After removing the segments containing events or glitches, as well as the edges of each segment, the final dataset consists of 265724 segments.

For each 1-second segment, a glitch was generated using a pair of parameters $\boldsymbol{\vartheta}$ drawn from uniform distribution, over the ranges $\boldsymbol{\alpha} \in [0.1, 2]$ and $\log_{10} \boldsymbol{A} \in [-22, -18]$, with $\boldsymbol{\alpha}$ starting at 0.1 to avoid numerical issues. The parameter $\boldsymbol{A}$ was chosen arbitrarily within a range considered plausible for interferometer data. The glitch injection was performed by centering the whitened pulse in the middle of each 1-s segment. The glitches were injected at the center of each segment to avoid truncation, allowing for a more thorough analysis.

4.5.2 Discussion of analysis results

Before moving on to the quantitative analysis, it is useful to show some Q-transform plots to compare the simulated glitches with those cataloged by Gravity Spy. Figure 4.6 displays the Q-transforms of some simulated glitches, and it can be seen that their shapes resemble those of the real glitches shown in figure 4.1. The parameters shown above the plots are not meant to represent the different morphologies, as different values can produce glitches that look visually very similar.

After a first qualitative look at how the simulated glitches behave, it is useful to compute the set of parameters $\boldsymbol{\lambda}$ for each generated glitch and analyze the resulting distributions. Figures 4.7 and 4.8 show the 2D histograms of the features extracted from the simulated glitches, plotted as a function of the corresponding generation parameters $\boldsymbol{\vartheta}$. From the plots of $\boldsymbol{\alpha}$, it is clear that many features show a strong spread when $\boldsymbol{\alpha} < 0.5$. This effect is especially visible for `max_amp`, `der`, `integ`, `Pinteg`, and `Peak`, which are computed in logarithmic scale. For higher values of $\boldsymbol{\alpha}$, the plots do not show a clear separation on $\boldsymbol{\lambda}$ parameters between different $\boldsymbol{\alpha}$ values, and so the glitch properties do not change in a noticeable way. The other features are not easy to interpret, probably because the noise often dominates over the glitches, making the results harder to read. In the parameters `fwhm`, `width`, and `bw`, distinct bands appear because the calculation looks for signal intersections at discrete values set by the sample rate. The histograms

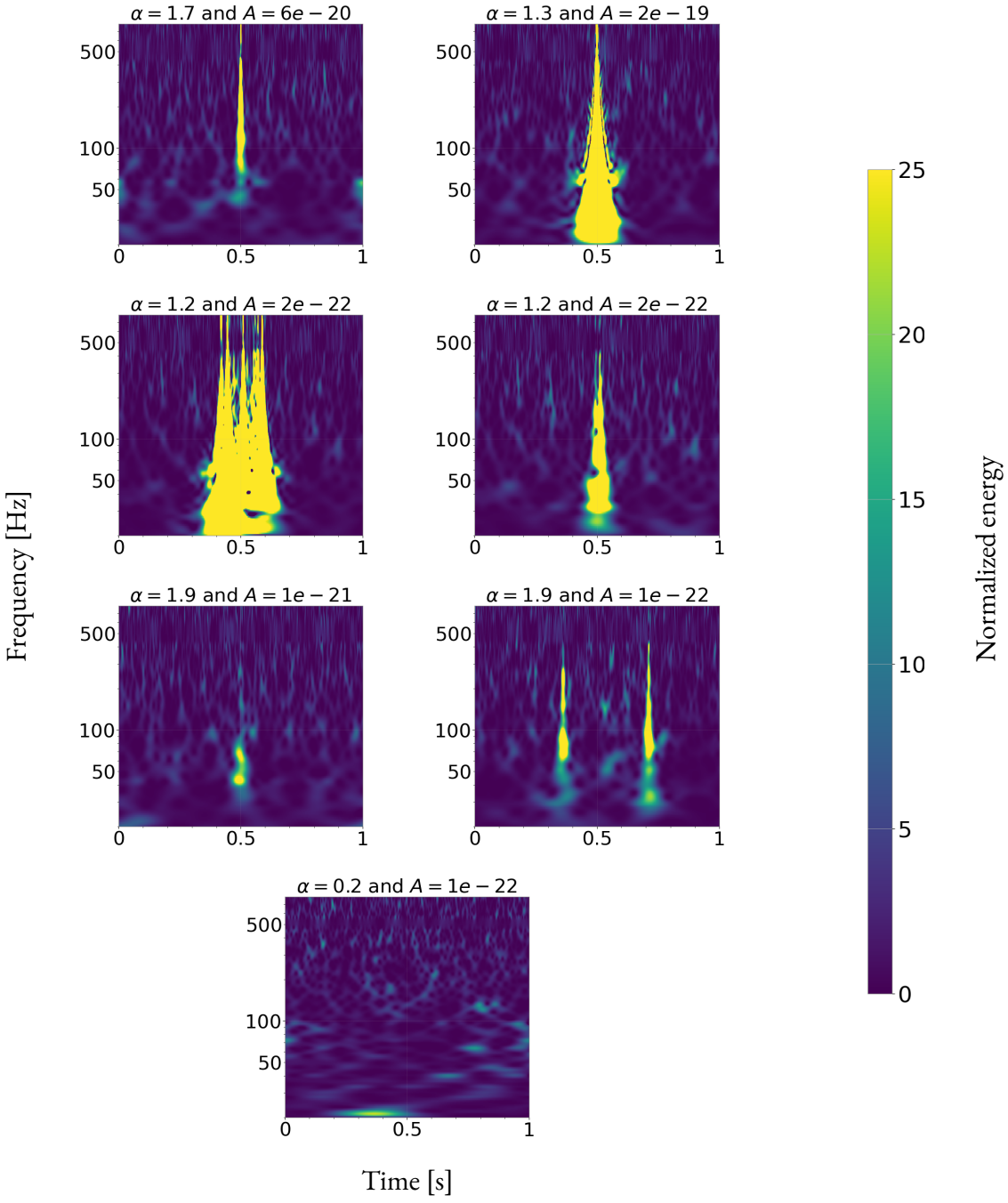


Figure 4.6: Q-transforms of some simulated glitches. Their shapes resemble those of glitches cataloged by Gravity Spy. The parameters shown above the plots are not meant to define distinct morphologies, as different parameter values can produce visually similar glitches.

in figure 4.8 indicate that most parameters vary linearly with $\log_{10} \boldsymbol{A}$. However, there is no variability at lower $\boldsymbol{A}$ values, probably due to the strong presence of noise at low amplitude scales, particularly for values below 10^{-21}.

4.6 Generation parameter inference

Analyzing the simulated glitches to produce realistic simulations requires estimating the generation parameters $\boldsymbol{\vartheta}$. Since the data are organized in 1-second time series, this problem is well suited for regression using a ML methods. Given the capability of NNs in analyzing time series [20], the first approach in this work was to train NNs on simulated glitches. The dataset consists of time series of glitches generated by drawing $\boldsymbol{\vartheta}$ parameters from uniform distributions, as described in the previous section. A portion of these examples is reserved for validation and testing, while the remaining ones are used for training. The regression task is defined to predict a pair of values corresponding to the two generation parameters. The architectures used were taken from [20] and adapted for regression tasks. Specifically, they consist of a standard CNN and an IT model. The networks were trained using the TensorFlow/Keras package [45, 46].

The main issue with this method concerns the difficulty of the network in distinguishing the parameter $\boldsymbol{\alpha}$. It was observed that, for $\boldsymbol{A}$, the network is able to capture its variations. This can be understood by looking at the distribution of $\boldsymbol{A}$ in figure 4.8, which likely reflects its linear dependence on many glitch $\boldsymbol{\lambda}$ features and shows that there is a certain amount of variability in the data that can be learned. For $\boldsymbol{\alpha}$, on the other hand, the variability decreases for values above 0.5 and becomes highly scattered below, as shown in Figure 4.7. This behavior probably confuses the network, which fails to identify average characteristics useful for correctly estimating the parameter.

Furthermore, it was noted that fixing $\boldsymbol{\alpha}$ values above 1 when generating the glitches, while leaving $\boldsymbol{A}$ free, improves the regression of $\boldsymbol{A}$, whereas fixing a specific value of $\boldsymbol{A}$ does not have a similar effect on $\boldsymbol{\alpha}$.

4.6.1 Regression of generation parameters with XGBoost

Since NNs did not yield the expected results, an alternative ML method based on XGBoost [21] was adopted to estimate the generation parameters $\boldsymbol{\vartheta}$ starting from the feature set $\boldsymbol{\lambda}$ introduced in the previous section. Unlike the direct approach used with NNs, which learn directly from

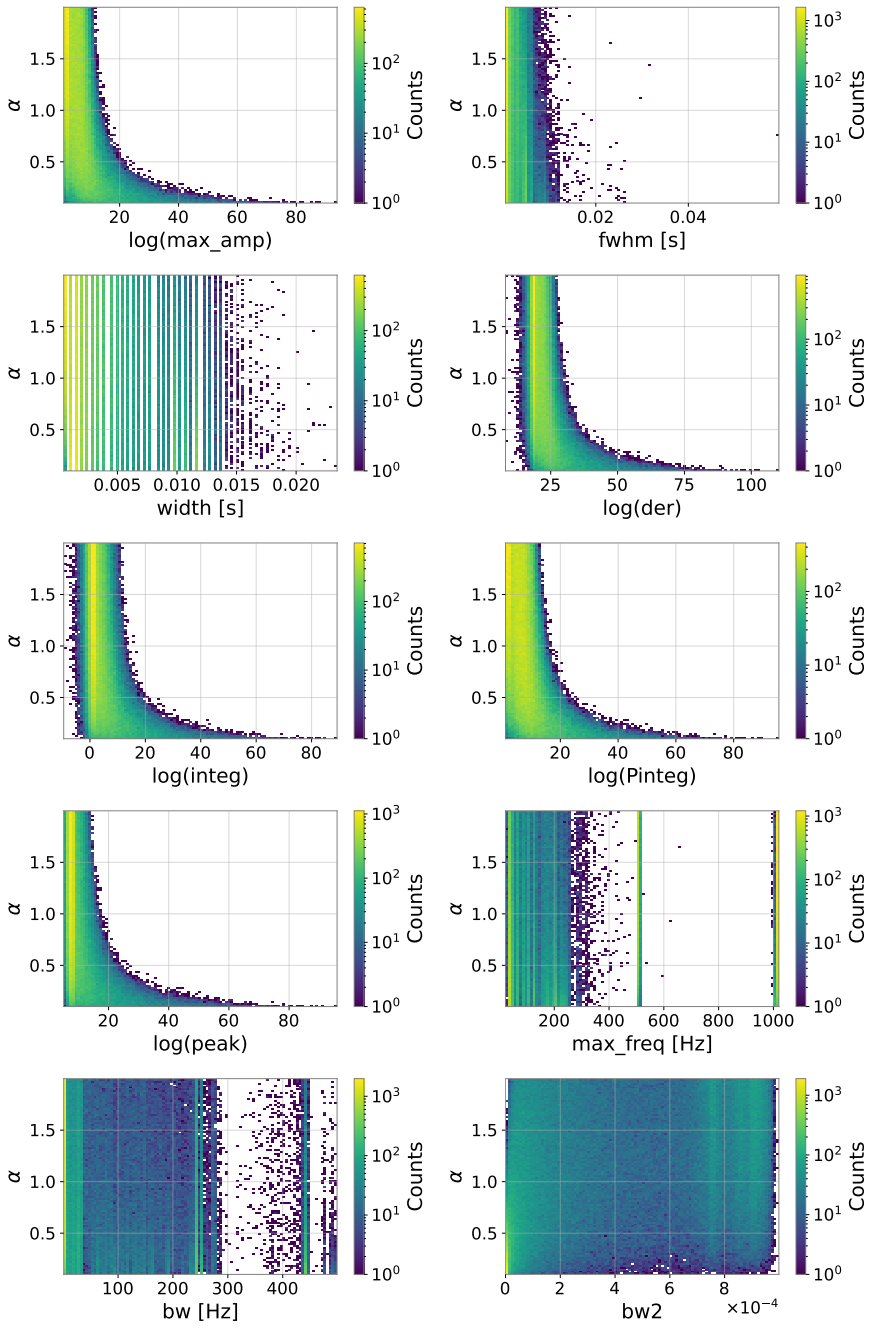


Figure 4.7: The graph shows two-dimensional histograms of the glitch features computed after the injection into the data, plotted as a function of α. For many parameters, a clear distinction can be observed between α values below and above 0.5. In particular, high α values correspond to distributions concentrated around well-defined parameter values, while low α values produce more widely spread distributions.

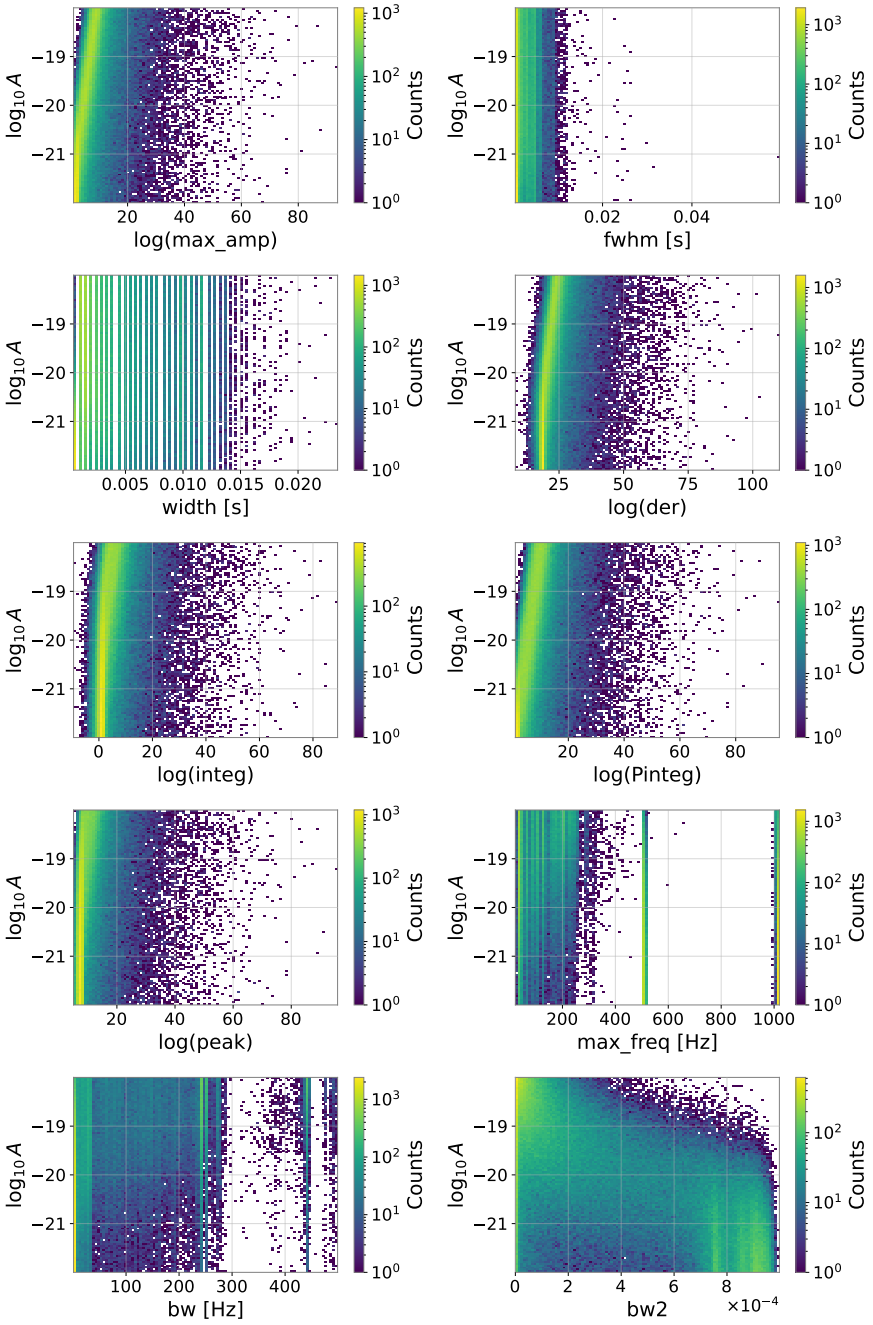


Figure 4.8: The graph shows two-dimensional histograms of the glitch features after injection into the data, plotted as a function of $\boldsymbol{A}$. For many parameters, the distribution of $\log_{10} \boldsymbol{A}$ values is approximately linear. For $\boldsymbol{A}$ values smaller than 10^{-21}, this linearity is lost, as the values are largely mixed with the noise.

the time series, this method relies on a set of features extracted from each glitch that summarize its main characteristics.

The dataset consists of the time series generated in the previous section. Consequently, the number of glitches used for training amounts to 265724.

The input matrix consists of the set of features $\boldsymbol{\lambda}$, and each feature set corresponds to the pair of parameters $\boldsymbol{\vartheta}$ that the model aims to predict. The dataset was first split into 80% for training and 20% for testing, to allow for an accurate evaluation of the model's performance. Then, 20% of the training set was set aside as a validation set.

Training architecture

For this work, two separate XGBoost regressors were trained, one for each generation parameter, since XGBoost does not natively support multi-label regression. Techniques such as the `MultiOutputRegressor()` from scikit-learn [47] can be used to train multiple labels simultaneously; however, in this case, each target is treated independently, resulting in a separate tree for each parameter. Some demos for multi-label regression exist, but they are still experimental [48]. The regressors were implemented using `XGBRegressor()` from the XGBoost package [49]. The predicted values have no intrinsic limits: the model can theoretically predict any real number. Unlike the true labels, which have a well-defined range determined by the dataset, predicted values can fall outside the range of observed labels.

For both models, the following settings were used: mean absolute error (MAE) as the evaluation metric, 1000 estimators, a learning rate of 0.1, maximum depth of 4, and partial row sampling (`subsample`) and column sampling (`colsample_bytree`) set to 0.9. Early stopping was set to 10 rounds based on the loss computed on the validation set.

Training and test results

Figure 4.9 shows the history of boosting rounds for MAE with respect to the two parameters, computed on both the training and validation sets.

Unlike $\boldsymbol{A}$, for $\boldsymbol{\alpha}$ a certain degree of overfitting is observed, as the validation error stabilizes while the training error continues to decrease. Early stopping was triggered when the validation set no longer showed improvements. The saved model for both parameters corresponds to the round with the best performance on the validation set.

The results obtained during the training phase are summarized in table 4.1.

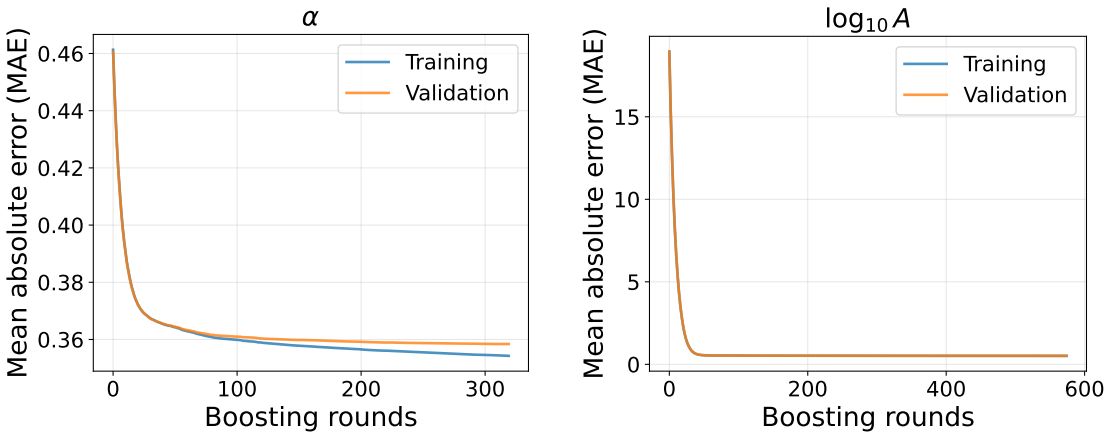


Figure 4.9: The graphs show the evolution of the MAE metric used to validate the regression during the boosting rounds for the two parameters ϑ. In the case of α, slight overfitting can be observed: the validation error tends to level off while the training error continues to decrease. This behavior is not present for parameter A, for which the two curves remain more consistent with each other.

Table 4.1: Best boosting round and corresponding MAE values for training and validation sets.

Parameter	Best round	MAE (train)	MAE (validation)
α	308	0.35	0.36
$\log_{10} A$	562	0.51	0.52

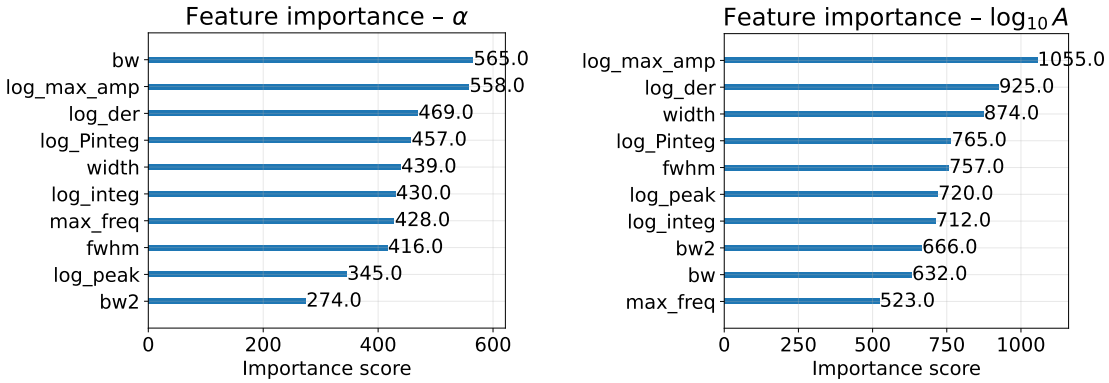


Figure 4.10: The graph shows the feature importance used for the regression with XGBoost, ranked by score: those for α are shown on the left, and those for A on the right. As can be seen, for both parameters, one of the dominant features is the maximum amplitude.

Figure 4.10 shows the feature importance plots obtained using `plot_importance()` function of XGBoost package [49].

The plots highlight which features had the greatest influence on the construction of the decision trees during training. In both cases, `max_amp` and the curvature of the peak (`der`) show the strongest influence on the prediction. Interestingly, the feature `bw`, which represents the spectral peak bandwidth, ranks as the most important for the regression of α.

During the test phase, the MAE for the two parameters was computed, resulting in a loss of 0.36 for α and 0.52 for $\log_{10} A$, which are consistent with the training results. These metrics were computed using the `sklearn.metrics` module from the scikit-learn package [47].

The test results are shown in figure 4.11, which compares the predicted values with the true ϑ parameters for glitches not seen during model training.

These plots include all predicted values. In particular, the axis limits are set so that the minimum and maximum on both axes correspond to the smallest and largest values found across both the predicted and true parameters.

For the parameter α, there is a concentration of data above 1 that is not accurately predicted and appears scattered across the true value range. However, the model seems to capture well the distinction between small and large α values. This may be due to the limited variability in the features λ above this threshold, as shown in the feature plots in Figure 4.7. Indeed, if the features show little variability for specific ranges of α, it is reasonable to expect that the model will also be unable to capture it.

Regarding A, the model shows a generally cleaner regression, although several outliers are still present. These outliers are partly explained by the use of MAE as a metric, which does not

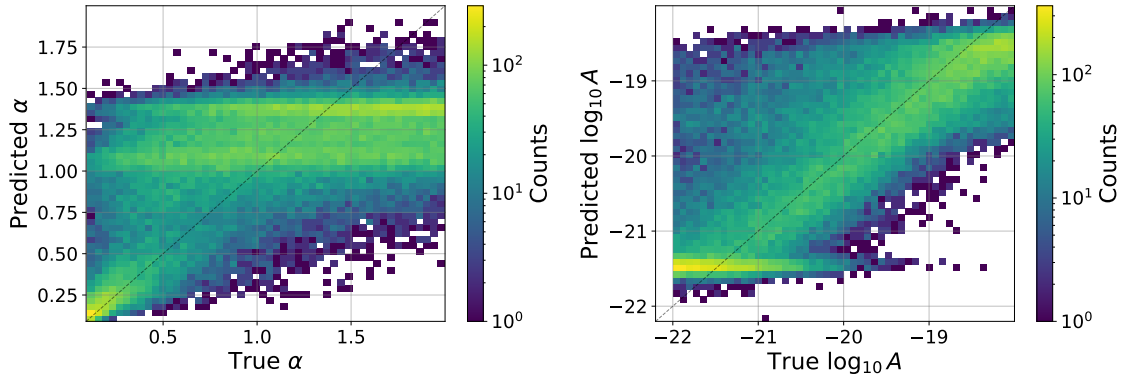


Figure 4.11: The plots show the distribution of predicted versus true ϑ parameters in the test dataset, after training the model with XGBoost. The dashed black line represents the ideal regression. For the parameter α, there is a concentration of data above ~ 1.25 that is not correctly predicted and scattered across the parameter space of true values larger than ~ 0.5. On the other hand, the model seems to capture well the distinction between small and large of ~ 0.5. Regarding A, the model shows a generally cleaner regression, although many outliers are still present, and values approaching 10^{-22} and 10^{-18} remain difficult to predict.

heavily penalize extreme deviations. Additionally, values approaching 10^{-22} and 10^{-18} remain difficult to predict. For values near 10^{-22}, this is likely because the signals are dominated by noise, whereas for values close to 10^{-18} the reason is not yet clear.

4.6.2 Mock test with simulated population

In this work, the goal is not to precisely determine the parameters ϑ of individual simulated glitches, but rather to reconstruct a population consistent with the target distribution. To this purpose, before estimating the parameters ϑ of the real glitches, it is useful to verify whether the XGBoost model can reconstruct the glitch population starting from a simulated population, for which the distributions of the measurable features λ are known.

The procedure is as follows:

1. Start from a simulated dataset of glitches with known parameters ϑ.
2. Compute the measurable features λ for each glitch.
3. Use the trained XGBoost model to predict the parameters from the simulated test population.
4. Compare the parameters λ of the reconstructed population with those of the original one to assess how well the model reproduces it.

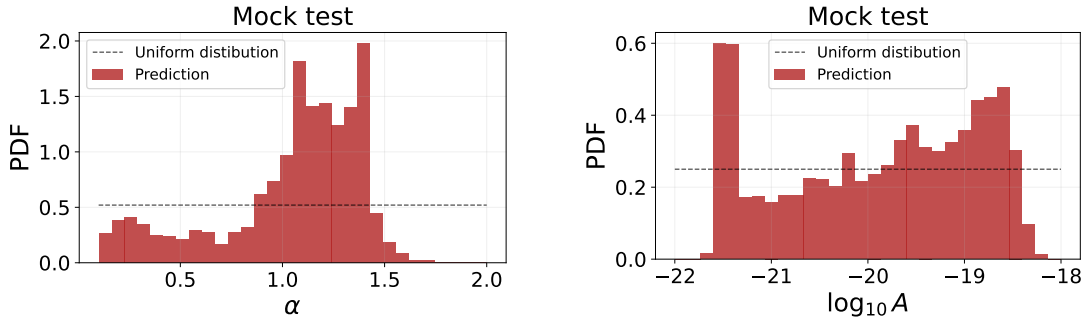


Figure 4.12: The figure shows the predicted PDFs for the parameters ϑ obtained from the model trained with XGBoost, compared with the theoretical uniform distribution. The predictions are generally accurate across most of the parameter space, but noticeable deviations from the theoretical PDF occur for high values of α and low and high values of A.

To simulate the target population, the same configuration as for the XGBoost training dataset was used, but with a different random seed to generate ϑ. Subsequently, 2500 segments were randomly selected from the simulated dataset for parameter estimation. This number of segments is chosen to be reasonably compatible with the available data for the real estimation, allowing the mock test to be performed with a sample size similar to that which will be used for estimating the real parameters.

Before showing the histograms of the measurable parameters λ, figure 4.12 reports the predictions of the parameters ϑ.

These plots were obtained using the `histogram()` function from the NumPy package [44], with `density=True`. This option normalizes the bar heights so that the histogram represents a PDF. Each bar's value is divided by the total number of counts and by the bin width.

It can be seen that, compared to the theoretical distribution, the parameters estimated are not perfectly uniformly distributed. This reflects the difficulties already encountered during the training phase with XGBoost. In particular, for α many values cluster around 1, while for A the model struggles to correctly predict the extreme values. The PDFs shown in the figure were thus used to generate glitches intended to reproduce the target population.

To highlight possible differences or similarities between the two populations, figure 4.13 shows the distributions of the features λ for both the target population and the reconstructed one.

The distributions of the simulated and predicted glitches are compared using histograms with Poisson error bars and a logarithmic y-axis, making the tails of the distributions visible. In the legend, "train" refers to the reconstructed population and "test" to the target popula-

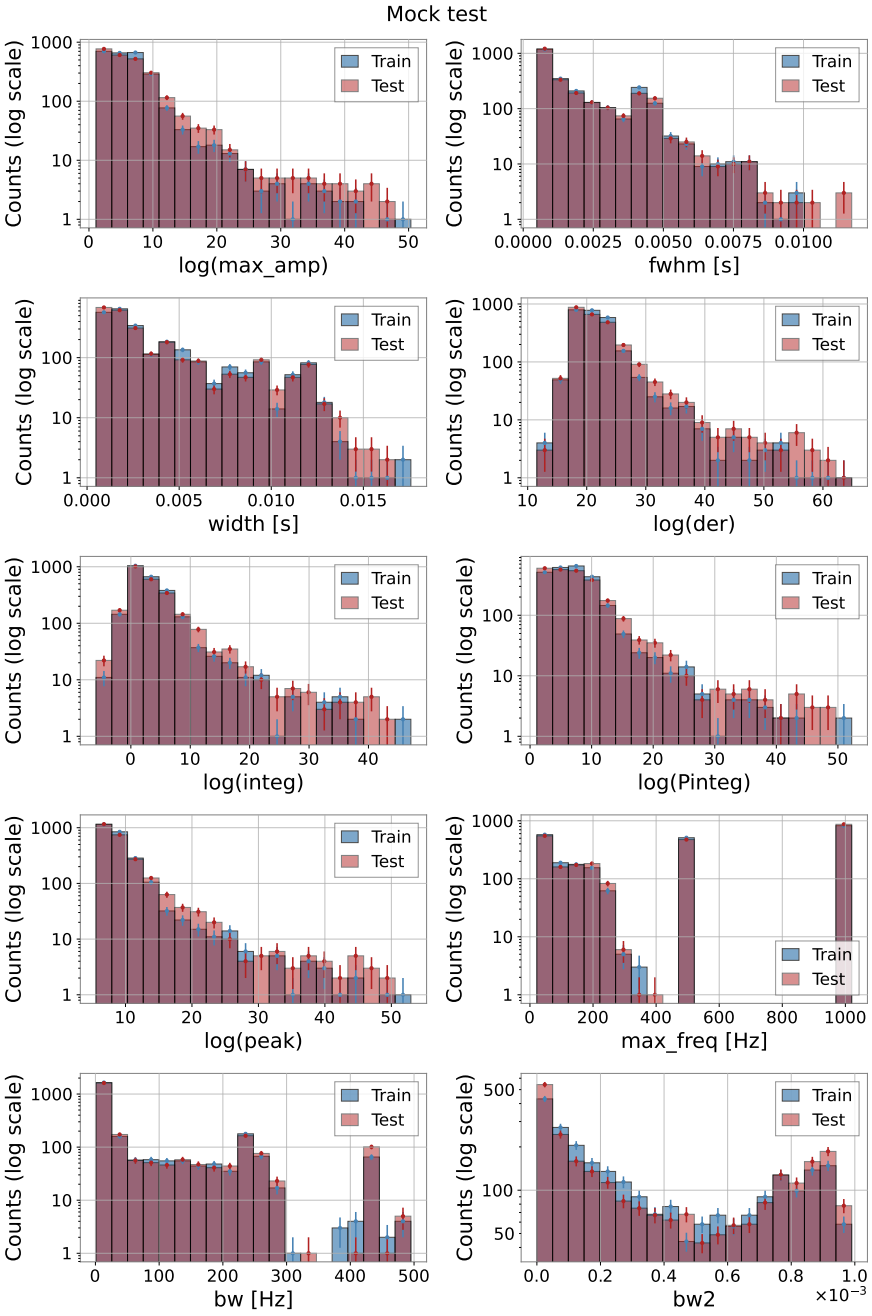


Figure 4.13: The figure shows a comparison between the histograms of the set of parameters λ calculated for the population of synthetic glitches with predicted ϑ (train) and the population of synthetic glitches with uniform distribution of ϑ (test). Poisson error bars are added to the histograms.

tion. The number of simulated glitches generated according to the ϑ parameter distributions matches the number of glitches used for testing, which were randomly drawn from the generated set to produce comparable histograms. The comparison shows that the reconstructed population closely matches the target distribution, indicating that the model is able to capture the general characteristics of the glitches.

4.6.3 Parameter inference on real glitches

As described previously, 2173 glitches were identified in the segments downloaded from GWOSC using the Gravity Spy catalog. These glitches were selected by choosing those for which the Gravity Spy ML model assigned a confidence greater than 90% that they belonged to one of the target classes [4]. These glitches constitute the population used for the inference of the parameters ϑ. The difference compared to the procedure described in section 4.5.1 is that the 1-second segments were selected so that each real glitch peak is centered. This choice is possible because the Gravity Spy catalog provides the peak time of each glitch with nanosecond precision [4]. Therefore, the segments can be centered on the peak, avoiding glitches being cut at the edges and ensuring analysis conditions comparable to those of the simulated case.

In figure 4.14, the results of the ϑ parameter estimation are shown.

From the distributions of the ϑ parameters, it can be seen that the values are strongly peaked around specific regions. This suggests that the glitch population shares similar properties ϑ. In the case of A, however, besides the main peak there is also a slight tail extending toward larger amplitude–scale values, which may indicate the presence of less common glitch classes.

The exact peak values of the α and A distributions are not relevant for our analysis, as these distributions are used only to generate a simulated population that reproduces the main features of the real one.

Figure 4.15 shows the histograms of the λ feature set. The glitches labeled as train data were generated from the distributions in 4.14. From the full simulated dataset, the same number of glitches used for the estimation of ϑ parameters were randomly selected. Consequently, as in the mock test, the number of glitches in the generated and target populations shown in 4.15 is the same. The plots show that the XGBoost model is able to reconstruct the glitch population with reasonable fidelity. Some minor differences can be observed in the tails of the distributions, indicating that the model reproduces the main characteristics of the population accurately, while rare extremes are less well captured. This behavior could be due to the fact that the dataset contains only 2173 glitches. Increasing the number of examples could help re-

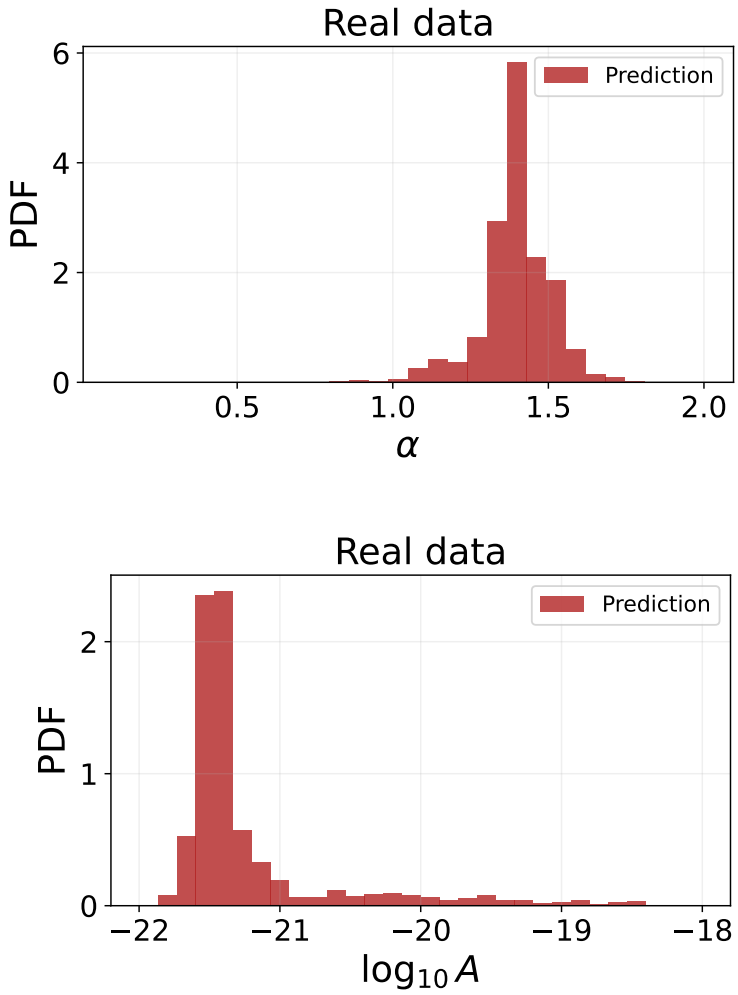


Figure 4.14: The figure shows the predicted PDFs for the parameters $\hat{\vartheta}$ of real glitches.

veal the presence of less common classes. Moreover, since the considered glitches come from just over three days of data, to improve statistics and obtain a more accurate estimate of the parameters ϑ, it would be necessary to download a longer time span and infer a much larger number of glitches, allowing the characteristics of rarer classes to become clearer.

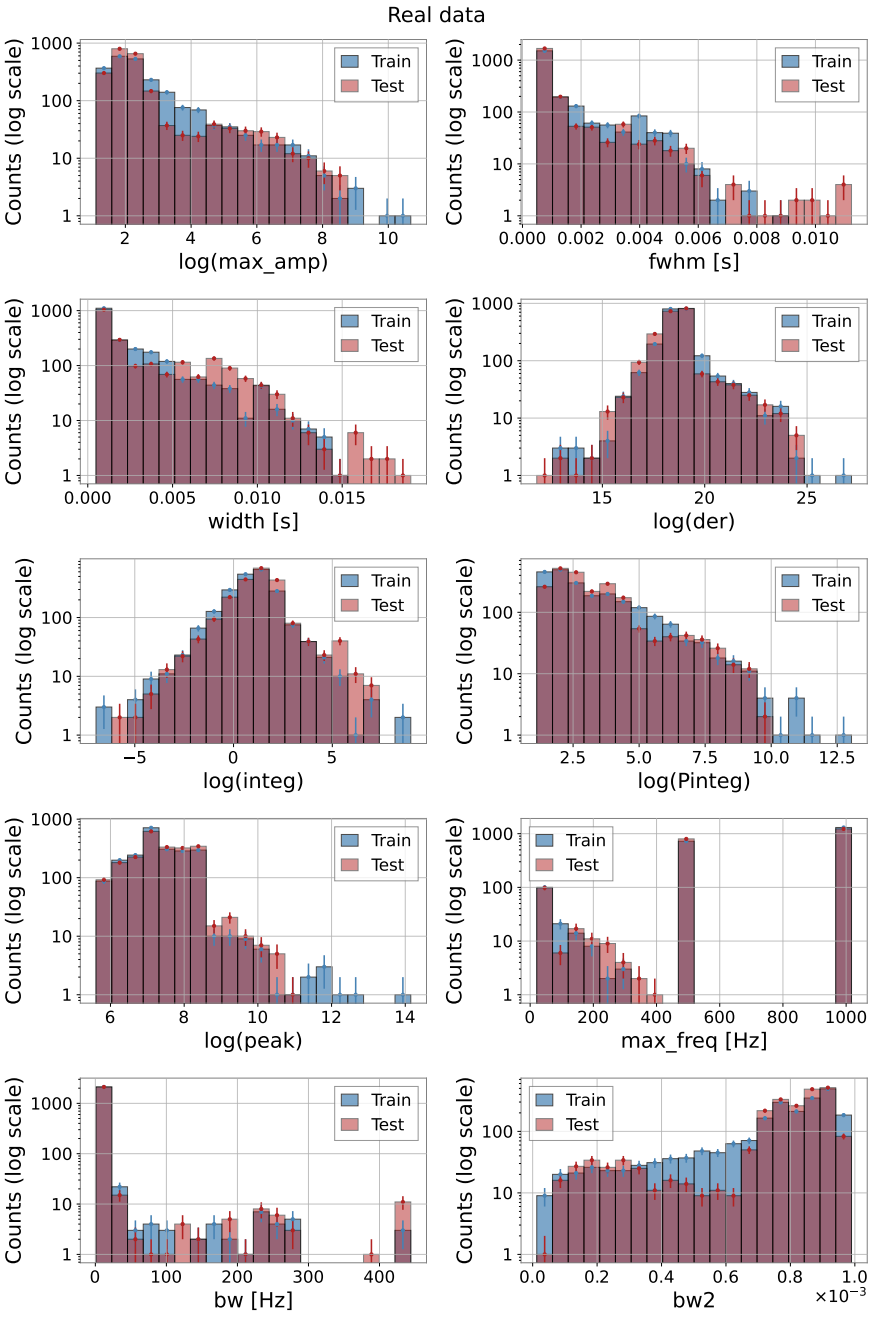


Figure 4.15: The figure compares the histograms of λ parameters for the population of synthetic glitches with predicted ϑ (train) and the population of real glitches (test). Poisson error bars are added to the histograms.

5

Classification of GW time series with NNs

The work presented in this thesis originates from the need to develop techniques capable of speeding up the online detection of GW signals, complementing traditional methods such as matched filtering. With the increasing detectors sensitivity and number of events observed during the O3 and O4a runs [1, 7], accelerating the detection process compared to traditional methods has become essential for the next observation runs. Moreover, having alternative techniques allows different detection methods to run in parallel and their results to be compared, which can be very useful.

It should also be noted that matched filtering has limitations in GPS segments where only one detector is active. By looking at the history of the different observing runs, it is possible to estimate how much observing time was spent with only one detector active. During O1, O2, and O4a, only the two LIGO interferometers were operating, and the single-detector periods comprised about 30% of the observing time. When Virgo joined the network in O3, this fraction decreased significantly, reaching about 15% in O3a and 11% in O3b. Overall, summing all single-detector periods from O1 to O4a, the total amounts to almost eight months of data [50]. As a result, there is a non-negligible amount of observing time during which only a single detector is active. These are periods in which the standard strategy to distinguish astrophysical signals from noise, based on temporal coincidence between multiple detectors, cannot be applied.

Chapter 3 introduced machine learning (ML) and discusses some techniques applicable to time series analysis. These ML strategies could complement more traditional approaches for

detecting GW signals in LVK strain data. In this context, the problem of identifying CBC signals has been addressed in [20] using neural networks (NNs).

Chapter 4 described a model capable of simulating several glitch classes. This model can be used to generate a dataset of simulated glitches, which helps train a NN classifier that learns to distinguish time series containing noise, glitches, and GW signals.

This chapter presents a pipeline developed to detect GW signals using NN classifiers, starting from strain data segments extracted from GWOSC and exploiting the glitch simulation process described in chapter 4. The results obtained with this approach are discussed.

5.1 Previous work

A common approach to GW detection using ML techniques is to formulate the task as a classification problem. The goal is to assess, for each fixed-length strain data segment, whether a GW signal is present or not. In [20], several NN architectures were tested to classify the LIGO Livingston detector data from the O1 run. The work consisted of splitting the LVK time series data into fixed-duration segments and training several NN architectures to learn how to distinguish these segments into three classes:

- `noise`: data compatible with a stationary noise background;
- `glitch`: data containing transient non-Gaussian noise artifacts (glitches);
- `signal`: data in which simulated astrophysical signals are injected into noise segments.

One challenge encountered in this work was that the number of glitch segments was much smaller compared to the noise and GW signal segments, resulting in an unbalanced dataset. This made it difficult for the network to learn the characteristics of the glitches, which were often misclassified, effectively reducing the problem to a two-class classification.

5.2 Use of the glitch model for data augmentation

In this context, the glitch model described in the previous chapter becomes useful for increasing the amount of training data. By using the model with the estimated generation parameters, it is possible to simulate realistic glitches and expand the dataset.

The following of this chapter presents the pipeline used to create the dataset of the three classes, which was then used to train a CNN classifier based on the approach described in [20].

This study does not aim to compare different architectures or to optimize performance by selecting the best model. For this reason, a single architecture based on [20] was adopted. The purpose is to assess whether the inclusion of simulated glitches in the dataset affects the behavior of the classifier and whether the model is able to correctly distinguish astrophysical signals from noise and glitches.

5.3 Generation of dataset for training and testing

The data for training and testing were extracted from the publicly released O3b observing run dataset via GWOSC [1]. In particular, the segments are the same as those used in chapter 4. Consequently, the time series are downsampled to 2048 Hz, whitened, high-pass filtered at 20 Hz, and divided into non-overlapping segments of 1 second. The following section describes how the three classes are constructed, and an example of each is shown in figure 5.1.

5.3.1 Noise class

The `noise` class consists of the data segments already used in chapter 4 for glitch injection. Consequently, these noise segments are predominantly stationary noise. The selection of segments was performed by removing the two known GW candidates [1] and the glitches listed in the Gravity Spy catalog [4]. The removed segments include not only the segment containing the peak but also the entire duration of the glitch, as specified in the catalog. It should be noted that Gravity Spy classified glitches with SNR > 7.5, so some lower-SNR glitches might still be present.

5.3.2 Glitch class

The `glitch` class consists of two subclasses, one of simulated glitches and one of real glitches, the first used for training and the second for testing. Simulated glitches are generated using the PDFs estimated in chapter 4 and injected into `noise` segments—i.e., without glitches listed in the Gravity Spy catalog and without GW signal candidates—making them compatible with the group of glitches cataloged by Gravity Spy. In this case, after the whitening step, each glitch is injected into `noise` data at a random position within the segment, ensuring that the glitch peak falls within the segment range. This is done because, unlike in chapter 4, where the goal was to analyze the simulated glitches and placing the peak at the center was useful, in this case

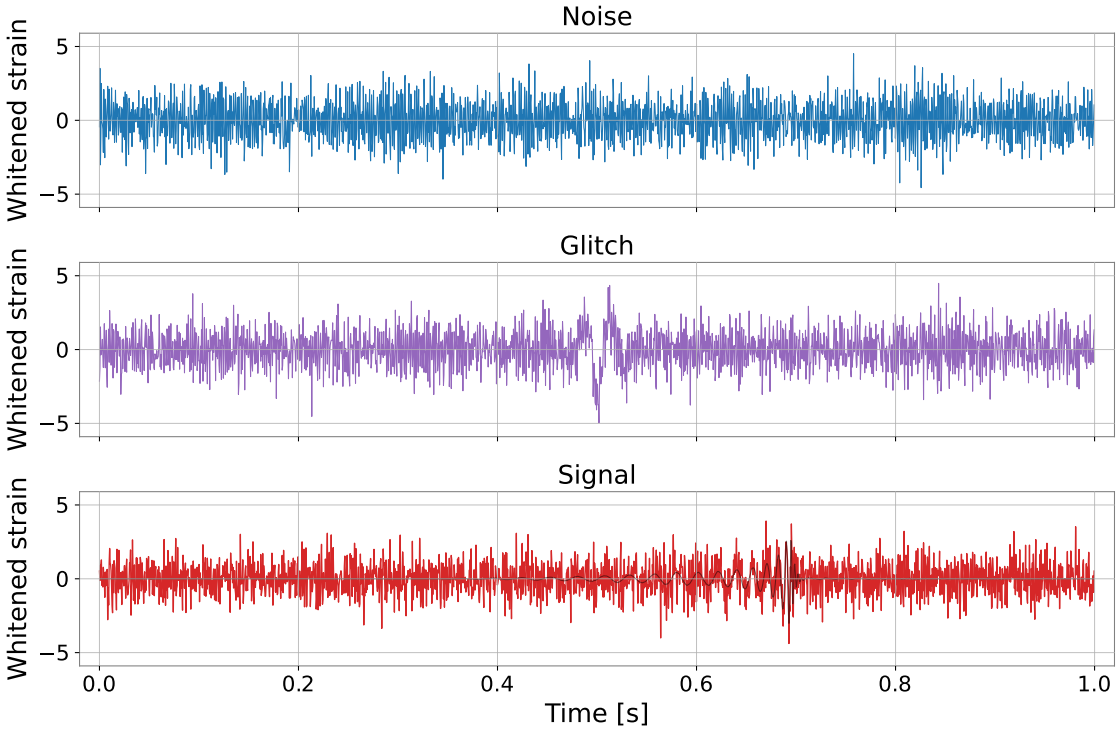


Figure 5.1: The plots show three examples of time series of noise (blue), glitches (purple), and signals (red). Each 1-second time series was taken from the LIGO Livingston detector during O3b, whitened, and high-pass filtered at 20 Hz. Top: `noise` segment at GPS 1256665621.5. Middle: `glitch` segment selected from Gravity Spy catalog [4] at GPS 1256666076.13. Bottom: `signal` segment into which a simulated BBH merger signal was injected, with an SNR of 20, at GPS time 1256666954.5. The black line highlights the waveform template injected into the strain data.

the glitch position must be random so that the segments more realistically reflect the variability of real data.

The second subclass includes real glitches from the Gravity Spy catalog with labels compatible with those that can be simulated. The glitch position within each 1-second segment is indirectly randomized, as the segments are cut without aligning to the glitch peak. As in chapter 4, the real glitches included in this class were selected from the Gravity Spy catalog, with an ML confidence higher than 90% [4]. Consequently, the dataset contains 2173 real glitches.

5.3.3 Signal class

The `signal` class was produced by taking the time series `noise`—i.e., without glitches listed in the Gravity Spy catalog and without real GW signal candidates—and injecting simulated CBC signals. To generate the astrophysical CBC signals, the approach described in [20] was followed. The waveform model used is `SEOBNRv4` [5] with a low-frequency cutoff at 30 Hz. This cutoff is applied because frequencies below this limit are dominated by noise and cannot be reliably detected by the instruments. These signals are then injected into noise segments to create the `signal` class. For this purpose, the simulated waveform templates are randomly shifted in time, ensuring that the merger occurs within the 1-second segment. They are then resampled at 2048 Hz, whitened using the same ASD as the `noise` segments into which they are injected, and high-pass filtered at 20 Hz.

The mass range of BBHs used to generate the CBC waveforms was chosen to be compatible with the range observed by LVK detectors and to ensure that the signals are fully contained within 1-second data segments. In particular, the component masses $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ were randomly drawn under the constraint that $\boldsymbol{m}_1 > \boldsymbol{m}_2 \geq 10\,\boldsymbol{M}_\odot$, while the total mass $\boldsymbol{M} = \boldsymbol{m}_1 + \boldsymbol{m}_2$ was uniformly distributed in $33\,\boldsymbol{M}_\odot \leq \boldsymbol{M} \leq 60\,\boldsymbol{M}_\odot$. this work considers non-spinning BH, so both spins are set to 0. The phase at coalescence and the polarization angle are drawn uniformly between 0 and $2\boldsymbol{\pi}$, while the inclination angle is drawn uniformly between 0 and $\boldsymbol{\pi}$. The right ascension and declination do not play a role in this study, since the analysis is carried out using a single detector. For this reason, both parameters are fixed to 0. The signal amplitude is set so that the resulting optimal SNR defined in equation 2.1 is uniformly distributed between 8 and 20. To achieve this, the source is initially placed at a distance of 100 Mpc, and then its amplitude is rescaled to obtain the desired SNR. Figure 5.2 shows the histograms of the luminosity distances, rescaled to produce a uniform SNR distribution for a small dataset of template waveform.

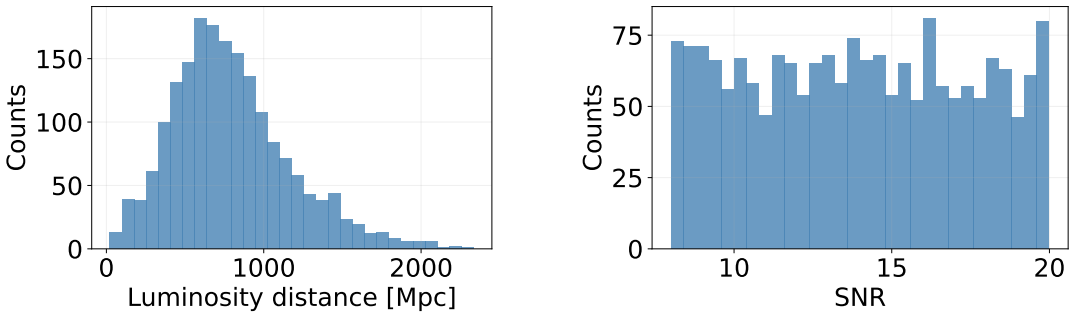


Figure 5.2: Histograms of the luminosity distances and the corresponding SNR for a subset of 1884 simulated CBC signals. The distances are rescaled to produce a uniform distribution of the SNR between 8 and 20.

Table 5.1: Architecture of the CNN used for time-series classification

Layer number	1	2	3	4	5
Type	Conv1D	Conv1D	Conv1D	Conv1D	Dense
Number of filters	256	128	64	64	—
Kernel size	16	8	8	4	—
Stride length	4	2	2	1	—
Activation function	`relu`	`relu`	`relu`	`relu`	`softmax`
Dropout rate	0.5	0.5	0.25	0.25	0.1
Max pooling	4	4	2	2	—

5.4 Classifier architecture

In this work, the NN is designed to classify time–series segments labeled with the three classes described in the previous section. The input to the model consists of vectors of length 2048, corresponding to 1-second time–series samples. Each vector is assigned the label of its corresponding class, and the model is trained to predict this label.

The NN model used in this work is the 1D-CNN selected in [20]. The architecture consists of five layers: four convolutional layers followed by a fully connected output layer. Each convolutional layer is characterized by its number of filters, kernel size, stride length, and activation function. After every convolutional block, max-pooling is applied for downsampling. The final layer is a fully connected layer that outputs the predicted class. Dropout rate is added for each layer. In this work, `same` padding was applied to the convolutional layers. Table 5.1 summarizes all the hyperparameters used to configure the model.

Table 5.2: Number of segments in each class of the dataset. The dataset is split into training and testing sets for `noise`, and `signal`, while the `glitch` (simulated) are used only in the training set and `glitch` (real) are used only in the testing set.

	`noise`	`glitch` (simulated)	`signal`	`glitch` (real)
Dataset	88574	88574	88574	2173
Training set	70860	88574	70860	–
Testing set	17714	–	17714	2173

5.5 Training and testing setup

Before training the model, it is necessary to properly prepare the dataset. The total number of downloaded segments amounts to 265724 seconds of data. These segments were divided into three equal parts, each assigned to one of the classes, ensuring that no GPS time is shared among them.

The dataset is divided in a training and a testing dataset. The numbers of 1-second segments for each class in these datasets is shown in table 5.2.

As a starting point, the `noise`, `glitch` (simulated), and `signal` classes contain the same number of segments. For the `noise` and `signal` classes, the data were split into a training set and a testing set with an 80/20 ratio. For the `glitch` class, all simulated glitches were used for training, while real glitches were reserved for testing. This is done to verify whether the inclusion of simulated glitches allows the model to correctly distinguish real glitches while still accurately classifying the `signal` class, which is our main objective.

Subsequently, the training set was further split into training and validation sets in an 80/20 ratio. These validation samples are used to compute the loss during the training phase. This is essential because computing the loss on the validation set helps the network generalize, improving its ability to correctly classify data not seen during training.

The networks were trained using the TensorFlow/Keras package [45, 46]. The loss used to optimize the classifier is the cross-entropy loss. The model is trained using the Adam optimizer with a batch size of 24 and a learning rate of 0.001. For training, early stopping was applied to stop the training process if the validation loss did not improve for 10 consecutive epochs. In addition to the loss, the accuracy metric is also computed to allow a direct comparison of the model's performance on the training and validation sets. this metric indicates the percentage of correct predictions [45, 46].

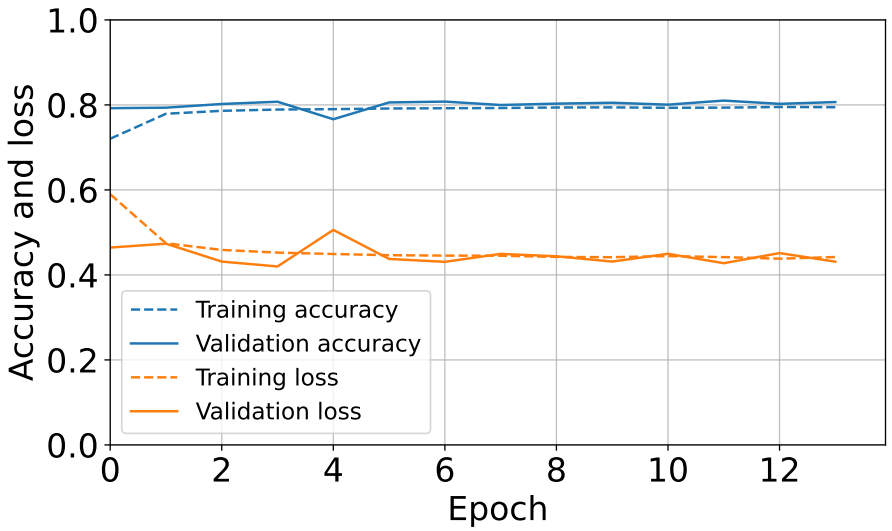


Figure 5.3: Evolution of the cross-entropy loss (labeled loss) and accuracy as the number of epochs increases. The training and validation curves remain close throughout the training, showing only slight fluctuations. As can be seen, the minimum of the validation loss is reached at the third epoch.

5.6 Results

This section presents the results of the training and subsequent testing of the CNN architecture used in this work. The training phase lasted for 13 epochs and was stopped early due to no improvement in the validation loss. Consequently, the best-performing epoch was epoch 3. In the figure 5.3, the training history is shown, illustrating how the loss and accuracy evolve over the epochs. It is noticeable that after only a few epochs the model reaches a plateau, and both loss and accuracy stop improving. This behavior may depend on several factors. One possibility is that the model actually reaches the minimum of the loss within the first few epochs. Another explanation is that the network might be trapped in a local minimum of the loss. These issues could be addressed by increasing the size of the training dataset, modifying the current architecture or optimizer parameters, or adopting deeper NNs. Another approach could be to remove early stopping and increase the number of training epochs, while monitoring the validation loss, to investigate whether the model performance can be further improved.

It can be observed that training and validation metrics, both for loss and accuracy, are consistent. During the testing phase, both the test loss and accuracy were computed. The results for the best epoch and for the testing phase are reported in table 5.3.

From this table, it can be seen that training and testing metrics are compatible, with even

Table 5.3: Performance metrics (loss and accuracy) of the CNN classifier on training, validation, and test sets.

	Training	Validation	Testing
Loss	0.44	0.43	0.39
Accuracy	0.79	0.81	0.93

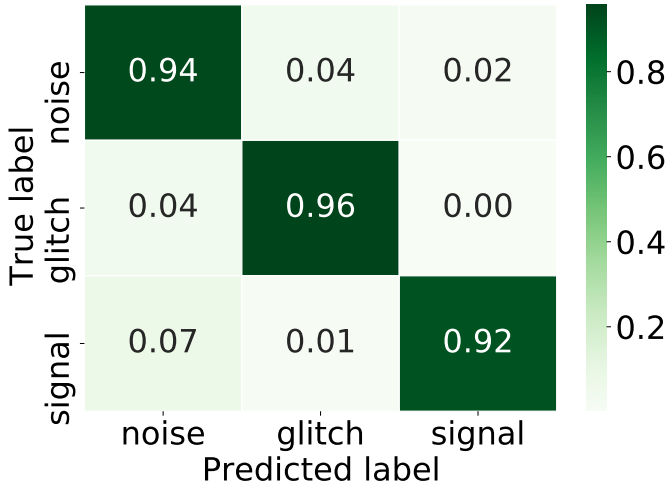


Figure 5.4: Normalized confusion matrix for the three classes. Each entry represents the fraction of samples in the test set assigned to each predicted class.

slightly better values on the test set, suggesting that the model generalizes well to unseen data.

During the testing phase, the model's performance was further evaluated using the confusion matrix, which shows the predicted versus true labels of unseen data. The confusion matrix was computed using the `sklearn.metrics` module of the scikit-learn package [47]. Figure 5.4 shows the confusion matrix, normalized by the number of samples in each class of the test set.

Unlike the training set, where the classes were balanced, the test set contains far fewer glitches, as these are only the real ones. Therefore, the confusion matrix is normalized to display the percentage of correctly classified versus misclassified samples. From the matrix, it can be seen that the testing phase demonstrates the model's ability to distinguish well among the three classes. Table 5.4 reports the confusion matrix in raw counts, showing the model's performance in terms of the number of samples correctly and incorrectly classified.

Since the main goal is the detection of astrophysical signals, an additional test was performed based on the computation of the receiver operating characteristic (ROC) [51] on `signal` class. This tool allows evaluating the performance of a classifier by showing its ability to correctly extract GW signals while rejecting noise and glitches.

Table 5.4: Confusion matrix in raw counts for the test set.

		Predicted label		
		Noise	Glitch	Signal
True label	Noise	16644	797	273
	Glitch	90	2080	3
	Signal	1301	204	16209

In particular, for each sample the output of the classifier is a vector with one probability per class, and the predicted label is chosen as the one with the highest probability. The ROC curve, instead, shows how the `signal` true positive rate varies with the false positive rate as the probability threshold is changed.

The classification efficiency (true positive rate) is defined as the fraction of `signal` segments correctly identified above the threshold, while the false alarm rate (false positive rate) is given by the fraction of `noise` and `glitch` samples incorrectly classified as `signal`. In this case, since each sample has a duration of 1 second, the false alarm rate is expressed in units of s^{-1}. The ROC curve was computed on the test set, i.e., the same data used to calculate the confusion matrix. By varying the probability threshold, the ROC curve shown in figure 5.5 is obtained. The ROC curve shows a classification efficiency of 80% at a false alarm rate of 10^{-4} s^{-1}, which corresponds to about 9 false alarms per day. To assess the model's performance at lower false alarm rates, it is necessary to evaluate it on a larger test set.

In addition, the area under the curve (AUC) was computed and found to be 0.99; for comparison, an AUC of 0.5 corresponds to random classification, while 1.0 indicates perfect classification.

As for the confusion matrix, both the ROC curve and the AUC were computed using the `sklearn.metrics` module of the scikit-learn package [47].

A few points are worth noting about glitch classification in this work. First, the simulated glitches used for training do not have a specific SNR threshold, unlike the Gravity Spy glitches, which are selected with SNR > 7.5. Despite this difference, the training performs well. Second, the glitch classes selected here are those most easily confused with astrophysical signals rather than with noise. This distinction is critical, as the network is primarily aimed at detecting GW signals. Only about 3 out of 2173 real glitches (approximately 0.14%) are misclassified as GW signals. Improving this number would require a larger test set of real glitches. Finally, in the previous work [20], the glitch class included all types of glitches, even those with very low SNR not cataloged by Gravity Spy, which contributed to confusion with the noise class. However,

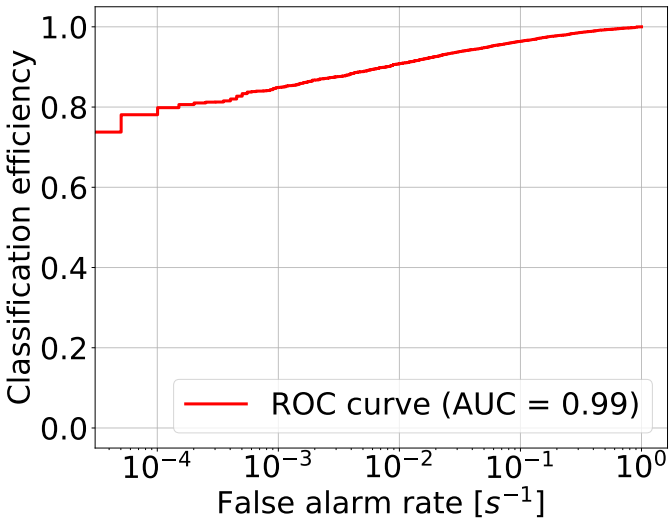


Figure 5.5: Receiver operating characteristic (ROC) curve for the `signal` class. The area under the curve (AUC) is also reported.

since the main goal is to detect astrophysical signals, it is more effective to train the network specifically to distinguish between glitches that resemble GW signals and the GW signal itself. This work follows that approach.

A possible extension of this work could be to expand the `noise` class to include all glitch segments that cannot be simulated by the model, perform the training using the same classifier architecture and the same training set of `glitch` and `signal` classes used in this work, and then evaluate the model's performance.

Conclusions

This section summarizes the main results of the thesis, highlights the key findings, and outlines potential directions for future research.

In this work, an alternative to traditional gravitational-wave detection methods, such as matched filtering, has been explored. The main idea is to use neural networks, which have shown good efficiency in recognizing astrophysical signals within the strain data of the LVK network. In this framework, the detection task is treated as a classification problem, where the goal is to determine whether a fixed-length data segment contains an astrophysical signal or not.

While many classification networks based on this principle have been developed in the last years, this work includes for the first time a simulation of glitches based on the Barkhausen noise hypothesis. This approach allows to increase the number of glitches available for training, improving the robustness of the model. The model for the glitch simulation requires parameters that are estimated by performing a regression with XGBoost from real glitches cataloged by Gravity Spy, ensuring realistic simulations consistent with observed data.

Using this glitch simulation model in a neural network-based classification pipeline demonstrates that it is possible to correctly classify a population of real glitches starting from a model trained on simulated glitches, without compromising the efficiency of gravitational-wave detection.

The classification results show that the network generalizes well to unseen data, correctly distinguishing background noise, glitches that most resemble astrophysical signals, and astrophysical signals themselves with high accuracy. In particular, only a few real glitches are misclassified as GW signals, confirming the effectiveness of the training strategy and the physical realism of the simulated glitches.

In conclusion, the proposed method for simulating a specific population of glitches can serve as a useful tool for gravitational-wave detection. In addition, this pipeline can be applied to other detectors, allowing for comparisons of estimated parameters and analyses of glitch characteristics. Future work could involve training with larger datasets and including other classes of glitches, and testing the network on periods of the run that were not used during training. This would allow us to check whether already known events are correctly identified and potentially discover new astrophysical signal candidates.

References

[1] R. Abbott and T. D. Abbott, "Gwtc-3: Compact binary coalescences observed by ligo and virgo during the second part of the third observing run," ***Phys. Rev. X***, vol. 13, p. 041039, Dec 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.13.041039

[2] D. M. Macleod, J. S. Areeda, S. B. Coughlin, T. J. Massinger, and A. L. Urban, "Gwpy: A python package for gravitational-wave astrophysics," ***SoftwareX***, vol. 13, p. 100657, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711021000029

[3] J. Glanzer, S. Banagiri, S. B. Coughlin, S. Soni, M. Zevin, C. P. L. Berry, O. Patane, S. Bahaadini, N. Rohani, K. Crowston, V. Kalogera, C. Østerlund, L. Trouille, and A. Katsaggelos, "Data quality up to the third observing run of advanced ligo: Gravity spy glitch classifications," ***Classical and Quantum Gravity***, vol. 40, no. 6, p. 065004, feb 2023. [Online]. Available: https://doi.org/10.1088/1361-6382/acb633

[4] J. Glanzer and S. Banagari, "Gravity spy machine learning classifications of ligo glitches from observing runs 01, 02, 03a, and 03b," Nov. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5649212

[5] A. Bohé, L. Shao, A. Taracchini, A. Buonanno, S. Babak, I. W. Harry, I. Hinder, S. Ossokine, M. Pürrer, V. Raymond, T. Chu, H. Fong, P. Kumar, H. P. Pfeiffer, M. Boyle, D. A. Hemberger, L. E. Kidder, G. Lovelace, M. A. Scheel, and B. Szilágyi, "Improved effective-one-body model of spinning, nonprecessing binary black holes for the era of gravitational-wave astrophysics with advanced detectors," ***Phys. Rev. D***, vol. 95, p. 044028, Feb 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.95.044028

[6] A. Nitz, I. Harry, and D. Brown, "gwastro/pycbc: v2.3.3 release of pycbc," Jan. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.10473621

[7] T. L. S. Collaboration, the Virgo Collaboration, and the KAGRA Collaboration, "Gwtc-4.0: Updating the gravitational-wave transient catalog with observations from the first part of the fourth ligo-virgo-kagra observing run," 2025. [Online]. Available: https://arxiv.org/abs/2508.18082

[8] B. P. Abbott, R. Abbott, and T. D. Abbott, "Gwtc-1: A gravitational-wave transient catalog of compact binary mergers observed by ligo and virgo during the first and second observing runs," ***Phys. Rev. X***, vol. 9, p. 031040, Sep 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.9.031040

[9] B. P. Abbott ***et al.***, "Gw150914: The advanced ligo detectors in the era of first discoveries," ***Phys. Rev. Lett.***, vol. 116, p. 131103, Mar 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.116.131103

[10] F. D. Guerra, "Studio di un modello fisico dei blip e altri glitch negli interferometri di onde gravitazionali," Tesi di laurea magistrale, Università degli studi di Trieste, Dipartimento di Fisica, 2024, supervisor: Prof. Edoardo Milotti.

[11] B. P. Abbott ***et al.***, "Observation of gravitational waves from a binary black hole merger," ***Phys. Rev. Lett.***, vol. 116, p. 061102, Feb 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.116.061102

[12] B. P. Abbott and R. Abbott, "The basic physics of the binary black hole merger gw150914," ***Annalen der Physik***, vol. 529, no. 1–2, Oct. 2016. [Online]. Available: http://dx.doi.org/10.1002/andp.201600209

[13] ——, "Gw170817: Observation of gravitational waves from a binary neutron star inspiral," ***Phys. Rev. Lett.***, vol. 119, p. 161101, Oct 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.119.161101

[14] ——, "Optically targeted search for gravitational waves emitted by core-collapse supernovae during the first and second observing runs of advanced ligo and advanced virgo," ***Phys. Rev. D***, vol. 101, p. 084002, Apr 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.101.084002

[15] P. Sutter, "Seeing the 'real' big bang through gravitational waves," July 2021, accessed: 2025-11-07. [Online]. Available: https://www.space.com/big-bang-study-with-gravitational-waves

[16] R. Abbott and T. D. Abbott, "Gwtc-2: Compact binary coalescences observed by ligo and virgo during the first half of the third observing run," ***Phys. Rev. X***, vol. 11, p. 021053, Jun 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.11.021053

[17] ——, "Gwtc-2.1: Deep extended catalog of compact binary coalescences observed by ligo and virgo during the first half of the third observing run," ***Phys. Rev. D***, vol. 109, p. 022001, Jan 2024. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.109.022001

[18] LIGO Scientific Collaboration and Virgo Collaboration and KAGRA Collaboration, "Open data from ligo, virgo, and kagra through the first part of the fourth observing run," 2025. [Online]. Available: https://arxiv.org/abs/2508.18079

[19] S. Alvarez-Lopez, A. Liyanage, J. Ding, R. Ng, and J. McIver, "Gspynettree: A signal-vs-glitch classifier for gravitational-wave event candidates," 2023. [Online]. Available: https://arxiv.org/abs/2304.09977

[20] A. Trovato, E. Chassande-Mottin, M. Bejger, R. Flamary, and N. Courty, "Neural network time-series classifiers for gravitational-wave searches in single-detector periods," ***Classical and Quantum Gravity***, vol. 41, no. 12, p. 125003, may 2024. [Online]. Available: https://dx.doi.org/10.1088/1361-6382/ad40f0

[21] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in ***Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining***, ser. KDD '16. ACM, Aug. 2016, p. 785–794. [Online]. Available: http://dx.doi.org/10.1145/2939672.2939785

[22] P. Mazzoldi, M. Nigro, and C. Voci, ***Fisica Vol 2: Elettromagnetismo - onde***, 2nd ed. Napoli: EdiSES, 1998.

[23] ——, ***Elementi di Fisica: Meccanica e Termodinamica***, 2nd ed. Napoli: EdiSES, 2008.

[24] J. B. Hartle, ***Gravity: An Introduction to Einstein's General Relativity***. Cambridge University Press, 2021.

[25] M. Maggiore, ***Gravitational Waves: Volume 1: Theory and Experiments***. Oxford University Press, 10 2007. [Online]. Available: https://doi.org/10.1093/acprof:oso/9780198570745.001.0001

[26] LIGO-Virgo-KAGRA Collaboration, "O3 spectral lines — gwosc," https://gwosc.org/O3/o3aspeclines/, accessed: 29 October 2025.

[27] K. Chatziioannou, T. Dent, M. Fishbach, F. Ohme, M. Pürrer, V. Raymond, and J. Veitch, "Compact binary coalescences: gravitational-wave astronomy with ground-based detectors," 2024. [Online]. Available: https://arxiv.org/abs/2409.02037

[28] M. Maggiore, ***Gravitational waves. Volume 2, Astrophysics and cosmology***, 1st ed. Oxford: Oxford University Press, 2018 - 2018.

[29] M. Wade, J. Betzwieser, D. Bhattacharjee, L. Dartez, E. Goetz, J. Kissel, L. Sun, A. Viets, M. Carney, E. Makelele, and L. Wade, "Toward low-latency, high-fidelity calibration of the ligo detectors with enhanced monitoring tools," ***Classical and Quantum Gravity***, vol. 42, no. 21, p. 215016, Oct. 2025. [Online]. Available: http://dx.doi.org/10.1088/1361-6382/ae1095

[30] P. Welch, "The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms," ***IEEE Transactions on Audio and Electroacoustics***, vol. 15, no. 2, pp. 70–73, 1967.

[31] B. P. Abbott and R. Abbott, "A guide to ligo–virgo detector noise and extraction of transient gravitational-wave signals," ***Classical and Quantum Gravity***, vol. 37, no. 5, p. 055002, feb 2020. [Online]. Available: https://doi.org/10.1088/1361-6382/ab685e

[32] S. Chatterji, L. Blackburn, G. Martin, and E. Katsavounidis, "Multiresolution techniques for the detection of gravitational-wave bursts," ***Classical and Quantum Gravity***, vol. 21, no. 20, p. S1809, sep 2004. [Online]. Available: https://doi.org/10.1088/0264-9381/21/20/024

[33] F. Robinet, N. Arnaud, N. Leroy, A. Lundgren, D. Macleod, and J. McIver, "Omicron: A tool to characterize transient noise in gravitational-wave detectors," ***SoftwareX***, vol. 12, p. 100620, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711020303332

[34] I. Goodfellow, Y. Bengio, and A. Courville, ***Deep Learning***. MIT Press, 2016, http://www.deeplearningbook.org.

[35] S. Shalev-Shwartz and S. Ben-David, ***Understanding Machine Learning: From Theory to Algorithms***. Cambridge University Press, 2014, free online copy available with permission from Cambridge University Press. [Online]. Available: https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/

[36] K. Team, "Adam optimizer — keras documentation," https://keras.io/api/optimizers/adam/, accessed: 2025-11-15.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[38] Keras Team, "Categoricalcrossentropy class – keras api documentation," https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class, 2025, accessed: 2025-11-14.

[39] X. Developers, "Introduction to boosted trees," https://xgboost.readthedocs.io/en/stable/tutorials/model.html, 2025, accessed: 2025-11-13.

[40] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," ***Data Mining and Knowledge Discovery***, vol. 34, no. 6, pp. 1936–1962, 2020. [Online]. Available: https://doi.org/10.1007/s10618-020-00710-y

[41] S. Soni, C. Austin, A. Effler, R. M. S. Schofield, G. González, V. V. Frolov, J. C. Driggers, A. Pele, A. L. Urban, G. Valdes, R. Abbott, C. Adams, R. X. Adhikari, and Ananyeva, "Reducing scattered light in ligo's third observing run," ***Classical and Quantum Gravity***, vol. 38, no. 2, p. 025016, dec 2020. [Online]. Available: https://doi.org/10.1088/1361-6382/abc906

[42] J. P. Sethna, K. A. Dahmen, and C. R. Myers, "Crackling noise," ***Nature***, vol. 410, no. 6825, pp. 242–250, 2001. [Online]. Available: https://doi.org/10.1038/35065675

[43] A. Baldassarri, F. Colaiori, and C. Castellano, "Average shape of a fluctuation: Universality in excursions of stochastic processes," ***Phys. Rev. Lett.***, vol. 90, p. 060601, Feb 2003. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.90.060601

[44] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," ***Nature***, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[45] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous systems," https://www.tensorflow.org/, 2015, software available from tensorflow.org.

[46] F. Chollet ***et al.***, "Keras," https://keras.io, 2015, software available from keras.io.

[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," ***Journal of Machine Learning Research***, vol. 12, pp. 2825–2830, 2011.

[48] XGBoost Developers, "A demo for multi-output regression," https://xgboost.readthedocs.io/en/stable/python/examples/multioutput_regression.html, 2025, accessed: 2025-11-25.

[49] Xgboost Developers, "Xgboost documentation," https://xgboost.readthedocs.io/en/stable/index.html, 2025, accessed: 2025-11-26.

[50] A. Trovato, E. Chassande-Mottin, M. Bejger, R. Flamary, and N. Courty, ***Neural Network Time-Series Classifiers for Gravitational-Wave Searches in Single-Detector Periods***. Singapore: Springer Nature Singapore, 2025, pp. 3–12. [Online]. Available: https://doi.org/10.1007/978-981-96-1737-1_1

[51] T. Fawcett, "An introduction to roc analysis," ***Pattern Recognition Letters***, vol. 27, no. 8, pp. 861–874, 2006, rOC Analysis in Pattern Recognition. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786550500303X

Ringraziamenti

Innanzitutto ringrazio il Prof. Marco Bazzan per la disponibilità e per l'opportunità che mi ha dato di svolgere la tesi con la collaborazione di Trieste.

Un grazie speciale va alla Prof.ssa Agata Trovato per il costante supporto, l'impegno a seguirmi e l'aiuto che ha reso possibile la stesura di questa tesi. Ringrazio il Prof. Edoardo Milotti per i numerosi spunti che hanno indirizzato questo lavoro. Inoltre, ringrazio Fabrizio del gruppo di Trieste per essere stato sempre disponibile e per il contributo che ha dato a questo progetto di tesi.

Ringrazio i miei colleghi astronomi per il tempo trascorso insieme, tra le lezioni e la mensa Murialdo. Siete stati spesso una spalla su cui contare e avete reso questi anni universitari pieni di bei ricordi. Grazie anche a casa di Linda e casa del Sorriso per avermi ospitato per numerose serate insieme.

Grazie a tutto il Gruppo non a Caso per il sostegno che ho sentito da tutti voi a finire questo percorso, per la vostra amicizia, la compagnia e le grigliate da Gio, ormai un ricordo.

Un ringraziamento speciale va al gruppo Via Turazza che è diventato un punto di riferimento per me dove mi sento sempre accolto.

Ringrazio i Paccovalli: siete compagni di mille avventure, mi avete aiutato in mille modi; mi siete sempre vicini anche se trovarsi al Paccovallo day ormai è un'impresa. Tutti voi continuate a essere una guida e uno stimolo. So che sono l'ultimo ma siate clementi.

Ringrazio i miei fratelli, Sofia e Leonardo, che tra le numerose pastasciutte preparate e la vostra vicinanza quotidiana, siete stati un prezioso sostegno in questo periodo. Ringrazio Zasha per contribuire a mantenere la serenità in casa anche nei momenti in cui sembra più difficile.

Un ringraziamento speciale va ai miei genitori, Annarosa e Roberto. In questi anni di studio mi avete sempre incoraggiato, mi avete dato fiducia senza mai farmi mancare nulla; è grazie a voi che ho fatto questo percorso e per questo a voi dedico questa tesi.

In conclusione, grazie a Carlotta. Non ci sono parole per ringraziarti: mi sei stata vicina in questo percorso altalenante, tra momenti di serenità e difficoltà. Hai portato molta pazienza, forse anche troppa, e sei stata una presenza importante per concludere questo percorso.