Mémoire

Auteur : Weidemann, Nell
Promoteur(s) : Fays, Maxime
Faculté : Faculté des Sciences
Diplôme : Master en sciences spatiales, à finalité approfondie
Année académique : 2023-2024
URI/URL : http://hdl.handle.net/2268.2/21212

Gravitational wave signal detection from core-collapse supernovae with the algorithm ALBUS

Weidemann Nell

Supervisor: Pr. Maxime Fays

Master in Space Sciences, focus: Research
Faculty of Sciences
Université de Liège

Academic year 2023-2024

Abstract

Gravitational waves are oscillations of spacetime itself and are produced by the most powerful and extreme events in the Universe. Predicted by the theory of general relativity, these waves were first detected in September 2015 by the LIGO interferometers, when two massive black holes merged, generating a spacetime deformation.

Other sources, such as core-collapse supernovae, are also believed to produce gravitational waves. The collapse of a massive star's core could generate signals that last less than a second, with waveforms that are not accurately known. As a result, traditional detection techniques, which rely on a good understanding of the targeted source's waveform, are ineffective. Deep learning techniques have been proposed as an alternative for detecting GW-generated power excess in time-frequency representations.

In this research project, we develop adaptations of the algorithm *ALBUS* for the detection of gravitational wave signals from core-collapse supernovae. ALBUS, which stands for Anomaly detection for Long-duration BUrst Searches, was originally designed for the detection of minute-long transient gravitational waves. It generates a time-frequency map highlighting pixels identified as potential GW signals. This work demonstrates that short-duration signal detection is possible by training a neural network algorithm, thereby opening up new possibilities for developing GW detection pipelines that leverage the speed and accuracy of neural networks.

This work is organized as follows: The first chapter introduces the concept of gravitational waves, the instruments for their detection, and the types of objects that emit them. It also provides details on the technical concepts employed in gravitational wave searches. The second chapter is dedicated to core-collapse supernovae, including the CCSN models of interest. The third chapter covers the fundamental concepts of deep learning, with an emphasis on convolutional neural networks. Chapter four details the adaptations made to ALBUS, including the results of network training and the integration of ALBUS into the pipeline *Pyxel*, along with background analysis results. Finally, chapter five outlines future work perspectives and potential applications.

Acknowledgments

Firstly, I would like to thank my supervisor, Mr. M. Fays, for his guidance and support throughout my master's study and research. I also wish to thank Mr. J-R. Cudell for his help throughout the year.

I would also like to extend my sincere thanks to Mr. J-R. Cudell, Mr. M-A. Dupret, and Mr. D. Sluse, for accepting to be members of my reading committee. I wish them an enjoyable reading.

Finally, I would like to thank my family and friends for their ongoing support and help throughout these last five years at university. A big thanks to the Didier in my life who have been a real support and source of happiness in everyday life.

Contents

Abstract i

Acknowledgments iii

1 Introduction to gravitational waves 1
- 1.1 Theoretical definition . . . 1
  - 1.1.1 General relativity . . . 1
  - 1.1.2 Gravitational wave emergence . . . 2
  - 1.1.3 Effect of passing . . . 3
- 1.2 Detector . . . 4
  - 1.2.1 Michelson interferometer . . . 4
  - 1.2.2 Modern interferometric detector design . . . 5
  - 1.2.3 Noise sources and sensitivity . . . 6
  - 1.2.4 First detection . . . 7
- 1.3 Gravitational wave sources . . . 7
- 1.4 Technical concepts . . . 9
  - 1.4.1 Antenna pattern . . . 10

2 Introduction to core-collapse supernovae 11
- 2.1 Outline of core-collapse supernovae . . . 11
- 2.2 Gravitational waves signal characteristics . . . 12
- 2.3 Waveform models . . . 12
  - 2.3.1 Models related to the neutrino-driven mechanism: . . . 12
  - 2.3.2 Models related to magneto-rotational mechanism: . . . 13
- 2.4 Gravitational waves searches from core-collapse supernovae . . . 14

3 Introduction to deep learning 17
- 3.1 Deep learning fundamentals . . . 17
  - 3.1.1 Artificial neural networks . . . 17
  - 3.1.2 Neural network training principle . . . 18
  - 3.1.3 Neural network training ingredients . . . 22
  - 3.1.4 Neural network training . . . 24
- 3.2 Convolutional neural networks . . . 26

3.2.1 Convolutional layer . . . 26
3.2.2 Pooling layer . . . 28

4 Anomaly detection for Short duration BUrst Searches **31**
4.1 ALBUS architecture . . . 32
4.2 ASBUS - ALBUS adaptations . . . 33
4.2.1 Time-frequency representation . . . 33
4.2.2 Target maps . . . 33
4.2.3 Neural network architecture . . . 34
4.3 Dataset . . . 36
4.4 Training . . . 38
4.5 Pyxel . . . 44
4.5.1 Background analysis . . . 45
4.5.2 Detection efficiency . . . 45

5 Future work and prospects **53**

Conclusion **55**

References **55**

A ASBUS dataset examples **61**

B ASBUS anomaly maps examples **65**

C Injection mask examples **69**

D Five loudest background triggers **71**

Acronyms

ALBUS	Anomaly detection for Long-duration BUrst Searches
AS	Anomaly Score
ASBUS	Anomaly detection for Short-duration BUrst Searches
ASD	Amplitude Spectral Density
CBC	Compact Binary Coalescence
CCSN	Core-Collapse SuperNova
CNN	Convolutional Neural Network
KAGRA	Kamioka GRavitational Wave Detector
LIGO	Laser Interferometer Gravitational-wave Observatory
ET	Einstein Telescope
GR	General Relativity
GW	Gravitational Wave
NN	Neural Network
PNS	Proto Neutron Star
PSD	Power Spectral Density
RGB	Red-Green-Blue
SASI	Standing-Accretion Shock Instability
SNR	Signal-to-Noise Ratio
TF	Time-Frequency

Chapter 1

Introduction to gravitational waves

1.1 Theoretical definition

1.1.1 General relativity

Gravitational Waves (GW) are a prediction of the General Relativity (GR) theory. They emerge as a solution of Einstein's equations under the weak gravitational field approximation. Before detailing their derivation, we define some general relativity concepts and equations.

Einstein's equations[1] are defined as

$$G_{\mu\nu} \equiv R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}. \tag{1.1}$$

The solution of these equations would provide the expression of the metric tensor $g_{\mu\nu}$. The metric tensor is fundamental as it describes the metric of the four-dimension space known as spacetime. In turn, the metric describes spacetime by allowing the definition of the distance notion. Let us break down this equation. The left-hand member, known as the Einstein tensor $G_{\mu\nu}$, is composed of the Ricci scalar

$$R = g^{\mu\nu} R_{\mu\nu}, \tag{1.2}$$

which is a contraction of the Ricci tensor

$$R_{\mu\nu} = R^{\alpha}{}_{\mu\alpha\nu} \tag{1.3}$$

being himself a contraction of the Riemann tensor $R^{\rho}{}_{\sigma\mu\nu}$. It is defined by the expression

$$R^{\rho}{}_{\sigma\mu\nu} = \partial_\mu \Gamma^{\rho}{}_{\nu\sigma} - \partial_\nu \Gamma^{\rho}{}_{\mu\sigma} + \Gamma^{\rho}{}_{\mu\lambda} \Gamma^{\lambda}{}_{\nu\sigma} - \Gamma^{\rho}{}_{\nu\lambda} \Gamma^{\lambda}{}_{\mu\sigma} \tag{1.4}$$

where the connections coefficients $\Gamma^{\alpha}{}_{\mu\nu}$, also called Christoffel symbols appears. They are defined as[2]

$$\Gamma^{\alpha}{}_{\mu\nu} = \frac{1}{2} g^{\alpha\beta} (\partial_\mu g_{\nu\beta} + \partial_\nu g_{\mu\beta} - \partial_\beta g_{\mu\nu}) \tag{1.5}$$

[1] We do not consider in this work the cosmological constant term $\Lambda g_{\mu\nu}$ added to account for the expansion of the Universe.

[2] This expression is valid if the coefficient is torsion-free ($\Gamma^{\alpha}{}_{\mu\nu} = \Gamma^{\alpha}{}_{\nu\mu}$) and if the metric is preserved at any point of spacetime.

The connection coefficients possess information related to curvature and allow distances to be measured in curved space. Concerning the Riemann tensor, it characterizes the curvature at any point in spacetime. The right-hand side of equation 1.1 contains the gravitational constant G, the speed of light c, and most notably, the stress-energy tensor $T_{\mu\nu}$. This tensor represents matter. Therefore, Einstein's equations link the Einstein tensor $G_{\mu\nu}$ and the energy-stress tensor; hence, relating the wrapping of spacetime to the presence of matter. The expression "matter tells spacetime how to curve, and spacetime tells matter how to move" takes on its full meaning. We invite the reader to refer to [1] for more details about those concepts.

1.1.2 Gravitational wave emergence

Now we can derive the gravitational waves from Einstein's equations under the weak field approximation [1, 2]. We consider a flat spacetime in which a small perturbation of the metric is introduced. The metric tensor $g_{\mu\nu}$ is written

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}, \qquad |h_{\mu\nu}| \ll 1 \tag{1.6}$$

with $\eta_{\mu\nu}$ the Minkowski metric tensor considered in special relativity, and $h_{\mu\nu}$ the metric tensor of the small perturbation. In this framework, we can linearize Einstein's equations with respect to $h_{\mu\nu}$ and find analytical solutions. We need to express Einstein's equation in the considered approximation, which is done by injecting expression 1.6 into equation 1.1. The Ricci tensor and Ricci scalar are also expressed with respect to eq. 1.6. By defining the tensor

$$\overline{h}_{\mu\nu} = h_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu}h \tag{1.7}$$

with $\overline{h} = -h$, and $h = \eta^{\mu\nu}h_{\mu\nu}$ and by considering the Lorenz gauge, *i.e.* $\partial^{\mu}\overline{h}_{\mu\nu} = 0$, we can derive the linearized Einstein's equations:

$$\Box\overline{h}_{\mu\nu} = -\frac{16\pi G}{c^4}T_{\mu\nu} \tag{1.8}$$

with $\Box = \partial_{\mu}\partial^{\mu}$ the d'Alembertian operator. With the absence of matter, the stress-energy tensor is null, and equation 1.8 writes

$$\Box\overline{h}_{\mu\nu} = 0. \tag{1.9}$$

In other words, gravitational waves correspond to a local perturbation that will propagate through the deformation of spacetime. This deformation will travel at the speed of light in vacuum, following the wave equation 1.9.

For instance, the monochromatic plane wave defined by the expression

$$\overline{h}_{\mu\nu} = A_{\mu\nu}\cos\Big(2\pi f(t - \frac{z}{c}) + \varphi\Big) \tag{1.10}$$

is a solution of eq. 1.9. The wave propagates in the z direction at a time t with an amplitude $A_{\mu\nu}$. The letters f and φ denote the frequency and the phase. The tensor $\overline{h}_{\mu\nu}$ is a 4×4 tensor with 16 components. However, $\overline{h}_{\mu\nu}$ is a symmetric tensor, reducing the number of independent components to 10. The Lorenz gauge further reduces this number to 6 independent components. By choosing an appropriate gauge, the degree of freedom of $\overline{h}_{\mu\nu}$ can be reduced to two (cf. [1]). This gauge is called the Transverse-Traceless gauge,

also noted as the TT-gauge. Therefore, equation 1.10 becomes in the TT-gauge

$$h_{\mu\nu}^{TT} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cos\Big(2\pi f(t - \frac{z}{c}) + \varphi\Big) \tag{1.11}$$

with $\overline{h}_{\mu\nu} = h_{\mu\nu}$. The wave is therefore transverse, *i.e.* the wave is perpendicular to its propagation direction, and the $h_{\mu\nu}$ trace is equal to zero. h_+ and $h_\times$ are the two GW polarizations: the plus and cross. A gravitational wave is said to be plus-polarized if $h_\times = 0$, and cross-polarized otherwise. If both polarizations are non-null, the gravitational wave is elliptically polarized. Now that we know what a gravitational wave is, we can ask ourselves how it impacts matter and how we can detect it.

1.1.3 Effect of passing

To understand the passing effect of gravitational waves on matter, we express the infinitesimal distance ds between two spacetime events in our previous approximation: we are in a flat spacetime in which a perturbation is present namely, a gravitational wave. For a chromatic plane wave, the metric tensor is written

$$\begin{aligned} g_{\mu\nu} &= \eta_{\mu\nu} + h_{\mu\nu}^{TT} \\ &= \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 + h_+ \cos\big(2\pi f(t - \frac{z}{c}) + \varphi\big) & h_\times \cos\big(2\pi f(t - \frac{z}{c}) + \varphi\big) & 0 \\ 0 & h_\times \cos\big(2\pi f(t - \frac{z}{c}) + \varphi\big) & 1 - h_+ \cos\big(2\pi f(t - \frac{z}{c}) + \varphi\big) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{1.12}$$

and the distance ds is characterized by

$$\begin{aligned} ds^2 &= g_{\mu\nu} dx^\mu dx^\nu \\ &= -c^2 dt^2 \\ &+ (1 + h_+ \cos\Big(2\pi f(t - \frac{z}{c}) + \varphi\Big))dx^2 \\ &+ (1 - h_+ \cos\Big(2\pi f(t - \frac{z}{c}) + \varphi\Big))dy^2 \\ &+ 2(h_\times \cos\Big(2\pi f(t - \frac{z}{c}) + \varphi\Big)dxdy \\ &+ dz^2 \end{aligned} \tag{1.13}$$

where $x^\mu = (t, x, y, z)$ are the spacetime coordinates. This equation can be easily interpreted by investigating the cases of plus- and cross-polarized gravitational waves. If $h_\times = 0$, then eq. 1.13 becomes

$$ds^2 = -c^2 dt^2 + (1 + h_+ \cos\Big(2\pi f(t - \frac{z}{c}) + \varphi\Big))dx^2 + (1 - h_+ \cos\Big(2\pi f(t - \frac{z}{c}) + \varphi\Big))dy^2 + dz^2. \tag{1.14}$$

Given the expression of the interval ds in Minkowski space, $ds^2 = -c^2 dt^2 + dx^2 + dy^2 + dz^2$, the only terms differing are the dx^2 and dy^2 terms. So, if a GW is present, the x direction is stretched by a quantity $\cos\big(2\pi f(t - \frac{z}{c}) + \varphi\big) > 0$, and the y direction is compressed by the same amount. Conversely, the x direction

is compressed and the y is stretched by the amount $\cos\big(2\pi f(t-\frac{z}{c})+\varphi\big) < 0$. It can be illustrated through the oscillation of a ring of matter that follows the spacetime deformation (cf. Figure 1.1 top panel). In turn, if $h_+ = 0$, we have

$$ds^2 = -c^2dt^2 + dx^2 + dy^2 + 2(h_\times \cos\Big(2\pi f(t-\frac{z}{c})+\varphi\Big)dxdy + dz^2. \tag{1.15}$$

To facilitate the discussion, we can define the x' and y' direction, shifted by 45° with respect to x and y axes. Equation 1.15 is in the new coordinate system

$$ds^2 = -c^2dt^2 + (1 + (h_\times \cos\Big(2\pi f(t-\frac{z}{c})+\varphi\Big))dx'^2 + (1 - (h_\times \cos\Big(2\pi f(t-\frac{z}{c})+\varphi\Big))dy'^2 + dz^2. \tag{1.16}$$

The spacetime distance ds undergoes the same deformation as for a plus-polarized GW along the x' and y' directions. The deformation undergone by a ring of matter in the presence of a cross-polarized GW is shown 1.1 bottom panel. The ring deformation sequence highlights a cross shape, from which stems from the name of the polarization. This is also true for the plus polarization. Note that gravitational waves are mostly a mixture of both polarizations, creating elliptically-polarized gravitational waves.

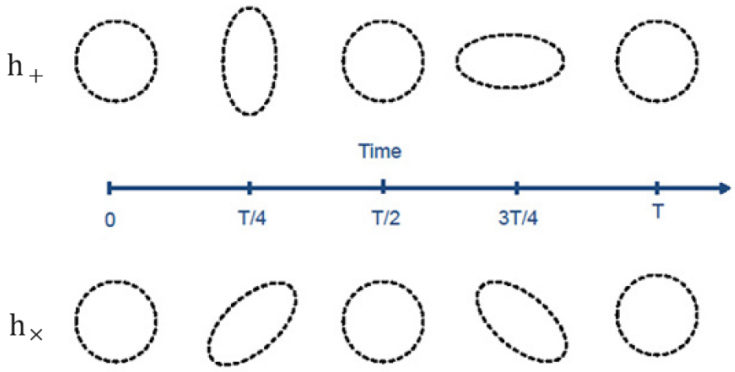

Figure 1.1: Illustration of the deformation undergone by a ring of matter over a period $T = 1/f$ in the presence of a plus-polarized (top panel) and a cross-polarized (bottom panel) gravitational wave [3].

1.2 Detector

The fundamental principles for GW detection involve their effect on matter. We have seen that a ring of matter is deformed along a given axis and the associated perpendicular direction. Thus, if we could detect the variation in length along these axes, we could detect passing gravitational waves.

1.2.1 Michelson interferometer

The experimental setting is based on the Michelson interferometer. Its design is such that a light beam is split into two perpendicular directions via a beam splitter, reflected then by mirrors along the incidence direction, and finally recombined at a photodetector. Interference patterns may emerge depending on the

phase shift between the two light beams [4]. Gravitational wave detectors are designed in such a way that the light beams recombine in opposite phases (under the condition that the lengths of their respective travel paths are equal). Destructive interferences are produced, and the photodetector records no light. In the presence of a gravitational wave, a differential arm motion is induced leading to light production. Therefore, modern interferometers measure the phase shift between two lasers induced by the variation of their travel distance along the detector arms. In practice, GW detection represents a true challenge since the order of magnitude of the strain (the time series measured by the interferometer) is about 10^{-22} for the strongest events. Therefore, the current interferometric detector design upgrades Michelson's interferometer through technical improvements [5].

1.2.2 Modern interferometric detector design

The first improvement is linked to the arms' length of the detectors. The relation $h = \Delta L/2L$ highlights that the longer the arms, the higher the sensitivity. The so called Fabry-Perot cavities increase the relative distance traveled by light. The principle is simple: additional mirrors are placed near the beam splitter so that the light is reflected multiple times before being recombined (about 300 times, *e.g.* increasing instrument arms' length from $4km$ to a relative distance of $1200km$ [5]). The second improvement artificially enhances the laser power up to the minimum required for GW detection. Indeed, The laser enters the instrument with a power of about ten watts, and it must reach hundreds of kilowatts to be able to detect gravitational waves [6]. Scientists introduced a power recycling mirror, which is a semi-transparent mirror placed between the light source and the beam splitter. It re-injects back into the interferometer the light that has already traveled in the cavities and has been reflected toward the source. Without this recycling, some power would be lost. Following the power recycling mirror idea, signal recycling mirrors are added to enhance the signal received by the photodetector. It increases further the laser power, increasing further the sensitivity. The Figure 1.2 outlines the current GW detector design including Fabry-Pérot cavities and recycling mirrors.

Ground-based detectors encounter several noise sources limiting their sensitivity. Hence, additional technologies have been developed to mitigate them. The most notable are the mirror suspensions and the ultra-high vacuum [8]. The mirrors and beam splitter are lifted via four suspensions. Leveraging the properties of a pendulum, this technique reduces any vibration that would induce disruptive motions. An active method is added to the ensemble, called the active isolation system. The mechanical structure associated with seismic sensors cancels out vibration by generating respective counter-motion. Besides, the interferometric detector operates in an ultra-high vacuum. It reduces the likelihood of interaction between air particles and photons coming from the laser. Such interaction may scatter light reducing the number of photons reaching the photodetector, or may emit light that would mimic or conceal a GW signal.

Several ground-based detectors have been constructed as a Michelson interferometer upgraded version during the past few years. The Laser Interferometer Gravitational-Wave Observatories (LIGO) are located in Hanford, Washington, and Livingston, Louisiana. Both have arms of a 4 km length. The Virgo interferometer has 3 km-long arms and is situated in Italy. The Japanese detector, namely the Kamioka Gravitational Wave Detector (KAGRA), joined the LIGO-Virgo collaboration searches in 2020, with underground arms of 3 km length. The four large detectors are involved in the active search for gravitational waves. Furthermore,

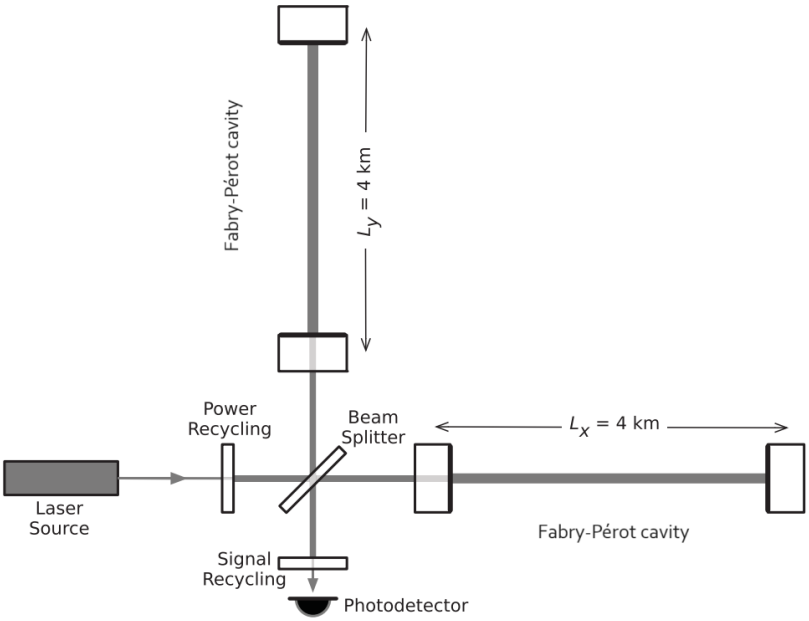

Figure 1.2: Diagram representing LIGO and Virgo interferometer design (adapted from [7]).

the interferometric detector GEO600, located in Germany, has developed and tested several technologies implemented in current large-scale instruments.

1.2.3 Noise sources and sensitivity

The sensitivity of the modern LIGO and Virgo interferometers is displayed in Figure 1.3. The overall sensitivity curve is defined by the combination of all noise source strains, resulting in a total strain noise outlined in the figure. The (total) noise strain translates the colored nature of the detector noise, meaning it varies in intensity across frequencies. Three main noise sources are at play: seismic noise, which is most significant below ~ 40 Hz (caused by ground vibrations, including earthquakes and human activities), thermal noise, which is dominant from ~ 40 Hz to ~ 200 Hz (due to thermal motion of atoms constituting the mirrors and their suspensions), and quantum noise dominant above ~ 200 Hz (due to discrete nature of light and the statistical uncertainty linked to photon counting performed by the photodetectors).

The noise strain presents additional features appearing as large spikes. These have various origins [9]. Notable features are the power line frequency (around 60 Hz in the US and 50 Hz in Europe), the resonance modes of the mirror suspension around $\sim$ 500 Hz and their harmonics (1000, 1500, 2000 Hz). They are known as the violin modes. Similar modes appear in the beam splitter suspension at about 300, 600, and 900 Hz. In addition, some of these noise signals are deliberately introduced to help calibrate and control the detectors.

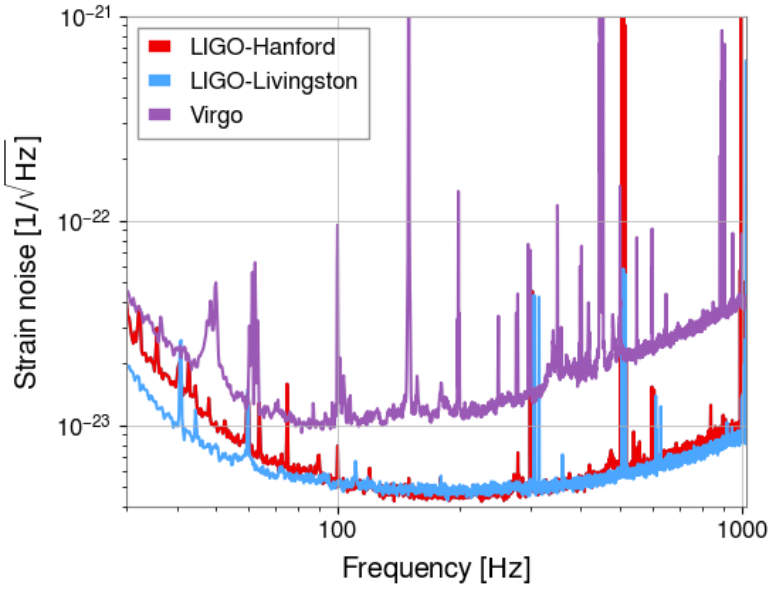

Figure 1.3: Sensitivity curve of LIGO and Virgo interferometric detectors during the third obeserving run.

1.2.4 First detection

The first gravitational wave detection, namely $GW150914$, occurred on September 14^{th}, 2015. The Advanced LIGO detectors detected the signal during the engineering run preceding the first observing run. The engineering run is part of the different phases for detector upgrades and sensitivity enhancements. During this phase, the engineering team conducts final tests to characterize the instrument and mitigate known noise sources as much as possible. It takes place before observing runs, which are phases where astrophysical results are expected to be produced. Further analysis supposed the GW source was a merger of black holes with 36 and 29 solar masses ($M_\odot$). The massive objects merged at one billion light years away from Earth and formed a 62 $M_\odot$ black hole. The energy carried away by the gravitational waves was about $3M_\odot$, and most of it was emitted in a fraction of a second. Remarkably, the GW power emitted was over ten times greater than the combined luminosity of all the stars and galaxies in the observable universe [7].

So far, the LIGO-Virgo collaboration has detected around 90 events originating from binary coalescences during the first three observing runs. The events are either black holes, neutron stars, or black hole-neutron star mergers. Furthermore, several other sources are expected to produce gravitational waves that have not been detected yet.

1.3 Gravitational wave sources

Currently, detectable gravitational waves are produced by the acceleration of massive objects. However, gravitational waves are produced under the condition of an accelerated non-spherically symmetric motion. Indeed, as a consequence of mass and momentum conservation, the first multipole contributing to the wave

amplitude is the source quadrupole moment. From eq. 1.8, the strain can be expressed in terms of the second derivative of the quadrupole moment [1]

$$h_{\mu\nu}^{TT} = \frac{1}{D}\frac{2G}{c^4}\ddot{Q}_{\mu\nu}^{TT}(t - D/c), \tag{1.17}$$

with D the distance to the source. The quadrupole is evaluated in the TT-gauge. The metric perturbation could also be written as a function of the two GW polarizations

$$h_{\mu\nu}^{TT} = h_+\mathbf{e}_+ + h_\times\mathbf{e}_\times \tag{1.18}$$

for $\mathbf{e}_+$ and $\mathbf{e}_\times$ the unit tensor of the plus and cross polarizations:

$$\mathbf{e}_+ = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad \mathbf{e}_\times \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{1.19}$$

In spherical coordinates h_+ and $h_\times$ writes [10]

$$h_+ = \frac{1}{D}\frac{2G}{c^4}(\ddot{Q}_{\theta\theta} - \ddot{Q}_{\phi\phi}), \qquad h_\times = \frac{1}{D}\frac{G}{c^4}\ddot{Q}_{\theta\phi}, \tag{1.20}$$

with

$$\begin{aligned}
\ddot{Q}_{\theta\theta} &= (\ddot{Q}_{11}\cos^2\phi + \ddot{Q}_{22}\sin^2\phi + 2\ddot{Q}_{12}sin\phi\cos\phi)\cos^2\theta + \ddot{Q}_{33}\sin^2\theta - 2(\ddot{Q}_{13}\cos\phi + \ddot{Q}_{23}\sin\phi)\sin\theta\cos\theta, \\
\ddot{Q}_{\phi\phi} &= \ddot{Q}_{11}\sin^2\phi + \ddot{Q}_{22}\cos^2\phi - 2\ddot{Q}_{12}\sin\phi\cos\phi, \\
\ddot{Q}_{\theta\phi} &= (\ddot{Q}_{22} - \ddot{Q}_{11})\cos\theta\sin\phi\cos\phi + \ddot{Q}_{12}\cos\theta(\cos^2\phi - \sin^2\phi) + \ddot{Q}_{13}\sin\theta\sin\phi - \ddot{Q}_{23}\sin\theta\cos\phi.
\end{aligned} \tag{1.21}$$

The strongest gravitational waves are produced by the most massive and fastest objects. Black holes and neutron stars are candidates for emitting gravitational waves. However, a single object moving along a rectilinear trajectory won't produce gravitational waves; they must be in a binary system. As they spiral, the emission of gravitational waves acts as a damping force. Their orbital angular frequency increases as they get closer until they finally merge. Once the resulting object stabilizes, spacetime perturbations cease. Binary systems composed of black holes, neutron stars, or both are part of the Compact Binary Coalescence (CBC) class. This first category encompasses sources producing gravitational waves with a modeled strain. It means that the process underlying the GW production is well studied and accurate models reproduce the source gravitational signature. The second-class sources are continuous gravitational waves and are also emitters of modeled-strain GW. The difference distinguishing the two categories is the signal duration. CBC generate short-duration GW (from seconds to several minutes), while continuous gravitational waves are emitted over a long period of time (their duration is at most the source lifetime). Continuous waves earned their names from the frequency and amplitude stability of the gravitational waves emitted by a single spinning object. For instance, a neutron star presenting a small upward imperfection enters the category. Additionally, the third and fourth classes include gravitational waves with non-modeled waveforms. Stochastic gravitational waves are an set of unresolved GW from every sky direction being mixed. The combination creates a stochastic

signal, which one could study statistically without precisely predicting it. The fourth category produces shorter signals than stochastic waves. This class is known as bursts. Due to the difference in treatment and challenges faced in burst searches, the class is further decomposed into short- (less than one second) and long-duration (above one second up to a few minutes) bursts. In this work, we focus on short-duration bursts, particularly on gravitational waves emitted by core-collapse supernovae. The next chapter will discuss these sources and their gravitational characteristics in detail.

1.4 Technical concepts

Gravitational wave searches employ many different techniques. We introduce some concepts involved in this work.

Gravitational wave signals are searched among interferometer noise over time. The data collected from gravitational wave detectors are time series that characterize the difference in length of the detector arms. After a calibration process, the detector output data is a strain $s(t)$ decomposed as a noise $n(t)$, and, if a gravitational wave passes through the Earth, a gravitation wave strain $h(t)$

$$s(t) = n(t) + h(t). \tag{1.22}$$

The signal can be represented in the frequency domain by computing the Fourier transform $\tilde{x}(f)$ defined by the general expression

$$\tilde{x}(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} dt. \tag{1.23}$$

This equation is true for continuous signals. In GW analysis, as the signal is a time series, it means it is sampled at regular intervals over time. Hence, the discrete Fourier transform is utilized and is written

$$\tilde{x}(t_k) = \sum_{j=0}^{N-1} x(t_j) e^{-i2\pi jk/N}, \tag{1.24}$$

with $x(t_j)$ evaluated at the time step $t_j = t_0 + j\Delta t$, $\Delta t = 1/f_s$ the time interval between two measurements defined by the sampling frequency f_s, and N the number of measurement samples.

We could also define the Power Spectral Density (PSD), quantifying the signal power at different frequencies [11],

$$S_x(f) = \frac{1}{N} |\tilde{x}(f)|^2. \tag{1.25}$$

The square root of the PSD is called the Amplitude Spectral Density (ASD). Both are commonly used to characterize the contribution of noise sources in the interferometer output.

Considering uncorrelated GW strain $h(t)$ and noise $n(t)$, one can show that a signal present in the strain s would produce an excess of power in $s(t)$ [12]. Therefore, a gravitational wave may be identifiable in time-frequency representation, including spectrograms. They represent a signal energy evolution over time and frequencies. They are generated by dividing successive time series intervals into segments and computing the PSD on each segment. The resulting PSDs are stacked along the y-axis and constitute a spectrogram

time bin.

1.4.1 Antenna pattern

The gravitational wave amplitude $h(t)$ measured by the detector depends on the source localization and can be expressed as a function of h_+ and $h_\times$ polarizations via

$$h = F_+(\theta, \phi, \psi)\, h_+ + F_\times(\theta, \phi, \psi)\, h_\times, \tag{1.26}$$

with F_+ and $F_\times$ the antenna pattern functions, and (θ, ϕ) the source position on the celestial sphere, *i.e.* θ denotes the declination and ϕ the right ascension. ψ corresponds to the polarization angle. The antenna pattern functions are defined in the equatorial coordinates by the expressions:

$$\begin{aligned} F_+ &= \frac{1}{2}(1 + \cos^2\theta)\cos 2\phi \cos 2\psi - \cos\theta \sin 2\phi \sin 2\psi, \\ F_\times &= \frac{1}{2}(1 + \cos^2\theta)\cos 2\phi \sin 2\psi + \cos\theta \sin 2\phi \cos 2\psi. \end{aligned} \tag{1.27}$$

The polarization angle is linked to the orientation of the source frame with respect to the specified detector frame. The antenna pattern characterized the angular sensitivity of detectors. The detectors are most sensitive to incidence angles perpendicular to the arms' plane and show a minimum sensitivity for GW propagating in the plane of the arms. As a consequence, constructing a network of detectors with different arm orientations would increase the sky coverage. The more networks are constructed the better the combined sensitivity over the sky is.

Chapter 2

Introduction to core-collapse supernovae

2.1 Outline of core-collapse supernovae

A basic scenario behind core-collapse supernovae (CCSN) can be described as follows: A massive star[1] in its final life stage burns heavy elements in successive shells, from hydrogen down to silicon, surrounding an iron core. This core becomes gravitationally unstable when a high enough temperature is reached, leading to the onset of collapse. During contraction, the core splits into two parts: the inner core and the outer core. The collapse is abruptly halted when the inner core reaches supranuclear densities and nuclear matter stiffens. The inner core's bounce produces a shock wave into the supersonically infalling outer core. As the shock wave progresses, it quickly loses energy and stalls. The shock wave needs to be revived by an energy deposit in the post-shock region for producing an explosion [13]. We consider the two main mechanisms currently investigated for powering CCSN explosions namely, the neutrino-driven mechanism and the magneto-rotational mechanism.

Slowly or non-rotating models are believed to be produced by neutrino-driven explosions, which represent the majority of explosions. As neutrinos carry most of CCSN explosion energy, a fraction is absorbed behind the shock. Standing-Accretion Shock Instabilities (SASI) may also develop enhancing neutrino heating powering the shock revival. The so-called SASI corresponds to non-radial oscillations of the shock. Magneto-rotational explosions may occur for rapidly rotating proto-neutron stars. The energy deposit comes from the rotational kinetic energy of the star core. In addition, strong magnetic fields enhance shock revival. This mechanism is thought to be less common than neutrino-driven explosions.

The CCSN explosion mechanism is still an active field of research. The processes involved are complex and detailing them is beyond the scope of this work. For further understanding, we refer to Abdikamalov *et al.* [13]. The next section outlines the GW generation within these mechanisms.

[1] Stars with a mass between 9 $M_\odot$ and 100 $M_\odot$

2.2 Gravitational waves signal characteristics

We know that gravitational waves are produced by compact objects presenting fast aspherical motions. Current simulations of CCSN suggest that most of the gravitational waves are emitted by the dynamics of the Proto Neutron Star (PNS), while a minor part is emitted by the asymmetric flows outside the PNS [13].

In not-rotating or slow-rotating models, the most common feature in spectrograms is a high-frequency arch-shaped component stemming from the PNS oscillations. It appears above 200 Hz, and can go up to $\sim$ 2000 Hz. A SASI signature also emerges in several models (*e.g. s25* model), corresponding to a low-frequency component (below $\sim$ 200 Hz). It lasts until the onset of shock expansion. For non-exploding models, the SASI component may last over a longer duration, as in $mesa20\_pert$ or NR models.

In fast-rotating models, a slower core collapse is expected along the equatorial plane than along the rotation axis due to the centrifugal force. Therefore, axisymmetric oblique perturbations are present at bounce time, which induces PNS oscillations. In addition, rotational non-axisymmetric instabilities may emerge and lead to a non-axisymmetric PNS shape. The rotation of non-axisymmetric objects would produce gravitational waves [13]. Rotation increases the amplitude of emitted gravitational waves compared to neutrino-driven CCSN and the combination of magnetic fields and rotation produces greater GW amplitudes. Hence, magneto-rotational explosions present larger detection distance ranges. In addition, a SASI component may appear, although with a lower contribution to the GW emission.

2.3 Waveform models

In this section, we list the waveform models used in this work. All the models are generated via $3D$ simulations. The frequency peak f_{peak} of the models is defined as the frequency at maximum energy dE_{GW}/df [14]. The peak frequency, translating the emission frequency at the time of the strongest emission, is an interesting quantity for detection prospects of gravitational waves from CCSN. The time given corresponds to the time after bounce. Besides, the models related to neutrino-driven explosions are all not-rotating. A time-frequency representation of each model is given in Figure 2.1 and Figure 2.2.

2.3.1 Models related to the neutrino-driven mechanism:

- Model *he3.5*:
  The progenitor at the origin of this exploding model is an ultra-stripped helium star with an initial mass of 3.5 $M_\odot$. Many stars evolve during their lifetime in a binary system and interact with their companion. Ultra-stripped stars are a result of these interactions, where the star has lost its outer layers via mass transfer and has evolved into a naked metallic core [15]. The waveform has been simulated by Powell & Müller [16]. The simulation stops 0.7s after the core-bounce time. The signal frequency peak is around 900 Hz.

- Model *s18*:
  This model has been developed by Powell & Müller [16]. The progenitor is a single star with a mass of 18 $M_\odot$ when the star first joins the main sequence *i.e.* when the fusion of hydrogen to helium begins in the stellar core. This mass is known as the Zero-Age Main Sequence (ZAMS) mass. As for *he3.5*, the GW signal develops with a peak frequency at $\sim$ 900 Hz. The simulation ends 0.89s after core bounce.

- Model *s13*:
  The progenitor is a 13 $M_\odot$ ZAMS star. The simulations is executed by Radice *et al.* [17]. The signal (stopped at 0.77s following the time bounce) has a peak frequency of ∼1400 Hz. This model does not lead to an explosion.

- Model *s25*:
  The simulations have been executed by Radice *et al.* [17] in the case of a 25 $M_\odot$ ZAMS star. It is stopped at 0.62s following the core-bounce time and the frequency peak is $\sim$ 1100 Hz.

- Model *SFHx*:
  Kuroda *et al.* [18] has probed the GW emission from a $15M_\odot$ progenitor. The GW signal ends $\sim$ 0.35 s after bounce and $f_{peak} \sim$ 700 Hz. No shock revival has been detected before the simulation end. Note that the *SFHx* evolution in TF maps should display an arcuate evolution. Our signal is folded horizontally from $\sim$ 500 Hz, resulting in a GW feature from 0.2 s to the end of the simulation between 200 and 400 Hz.

- Model *D15-3D*:
  Mezzacappa *et al.* [19] considered a 15 $M_\odot$ ZAMS star. The signal develops up to $\sim$ 2000 Hz before the end of the simulation. The shock is revived at $\sim$ 0.5 s.

- Model *mesa20_pert*:
  O'Connor & Couch [20] simulated a waveform with a frequency at maximum energy around 1000 Hz. The simulation is stopped after 0.53 s without leading to a shock revival. The progenitor star has a mass of 20 $M_\odot$.

- Model *Pan_2020_NR*:
  Pan *et al.* [21] model does not explode and forms a black hole at $\sim$ 0.78 s. The progenitor is a 40 $M_\odot$ star. The simulation lasts above 1 second and reaches frequencies above 200 Hz.

2.3.2 Models related to magneto-rotational mechanism:

- Model *m39*:
  m39 [22] corresponds to a Wolf-Rayet star characterized by a rapid rotation with an initial 600 km/s rotation velocity of its surface. The latter results from the evolution of a 39 $M_\odot$ helium star. The GW signal lasts for $\sim 0.55s$ after bounce and has a peak frequency of $sim670$ Hz. The model leads to an explosion around 0.2 s.

- Model *s15fr*:
  The Andresen *et al.* [23] model corresponds to a fast rotating model with an initial angular velocity Ω_0 equal to 0.5 rad/s. The model evolved from a 15 $M_\odot$ progenitor to produce a GW waveform with a $f_{peak} \sim$ 690 Hz. The shock is revived 0.26 s before the simulation stoppage at $\sim$ 0.46 s.

- Model *Pan_2020_SR*:
  Pan *et al.* [21] developed a 40 $M_\odot$ model characterized by slow rotation ($\Omega_0 = 0.5$ rad/s). The model produces an explosion 0.18 s prior to a black hole formation at $\sim$ 0.93 s.

- Model *Pan_2020_FR*:
  Pan *et al.* [21] also investigated a fast rotating model ($Omega_0$ = 1.0 rad/sec) for a progenitor of $40M_\odot$. The shock is revived around 0.25 s. The model does not form a black hole before the end of the simulation, *i.e.* within $\sim$ 0.46 s.

- Model *B12*:
  This 39 $M_\odot$ progenitor star model (simulated by Powell *et al.* [24]) has a frequency band going up to $\sim$ 2000 Hz with a frequency at peak amplitude of $\sim$ 1400 Hz. This rapidly rotating model is characterized by an initial surface rotational velocity of 600 km/s. The model is also characterized by a 10^{12}G magnetic field.

2.4 Gravitational waves searches from core-collapse supernovae

The search for gravitational waves from core-collapse supernovae is currently performed without any assumptions about the signal morphology. Processes involved in CCSN are stochastic, which means CCSN gravitational wave strains possess a stochastic nature. As a result, the standard detection technique called matched filtering techniques, which relies on a good knowledge of the GW strain of the targeted sources, is not suitable. Besides, only a limited number (around a hundred) of CCSN waveforms are currently available due to the simulation's expensive cost, increasing the need for alternative detection techniques. Nonetheless, the time-frequency evolution of CCSN gravitational waves holds recurrent patterns, which can be identified. Knowing the fact that neural networks present high performance for computer vision tasks (*e.g.* image classification, object detection, image segmentation, ...), the use of deep learning techniques has been proposed. We develop in the next chapter the fundamentals of deep learning and introduce neural networks, underlying the detection algorithm used in this work.

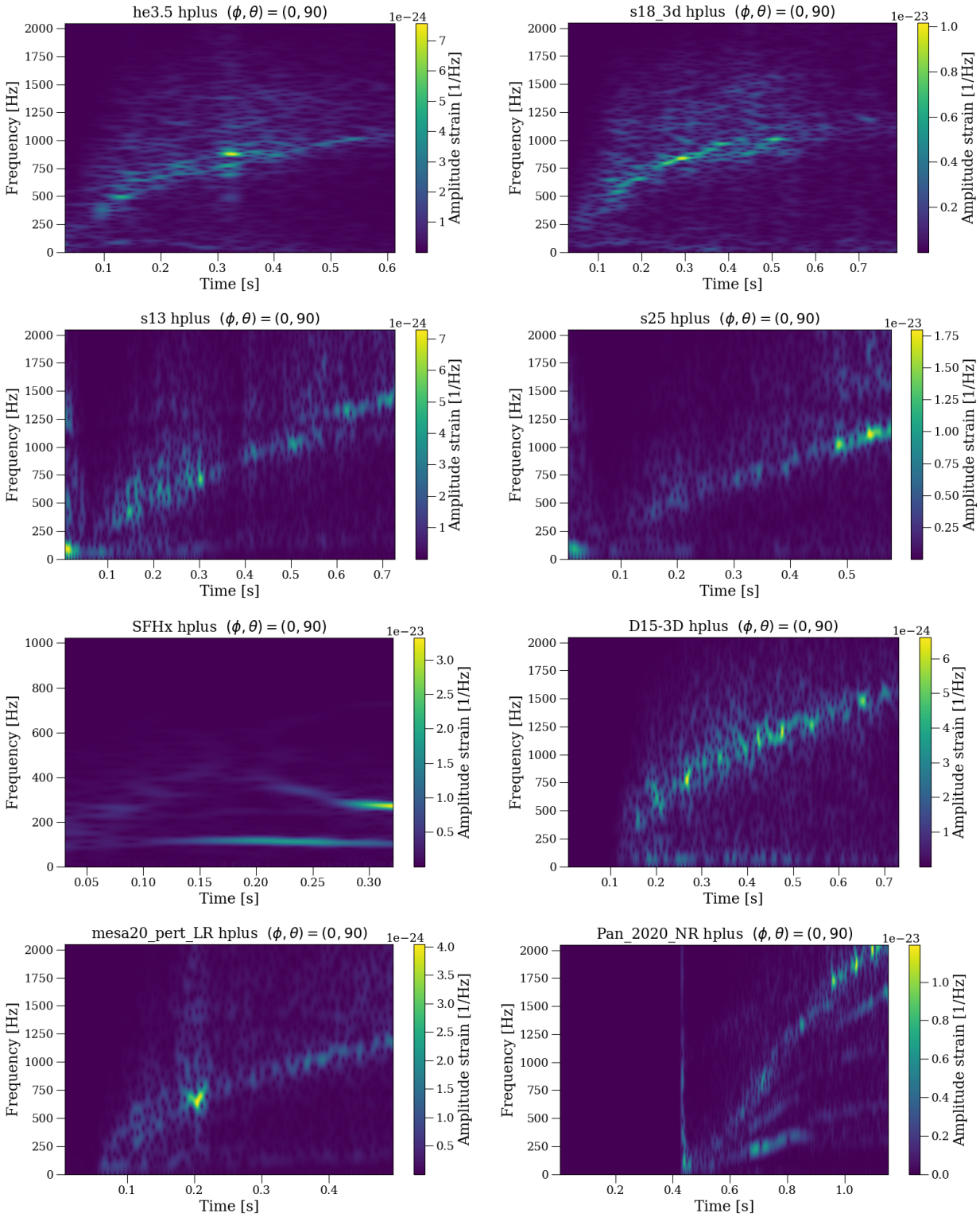

Figure 2.1: Time frequency representation of neutrino-driven model waveforms. The gravitational wave amplitude corresponds to a source distance of 10kpc.

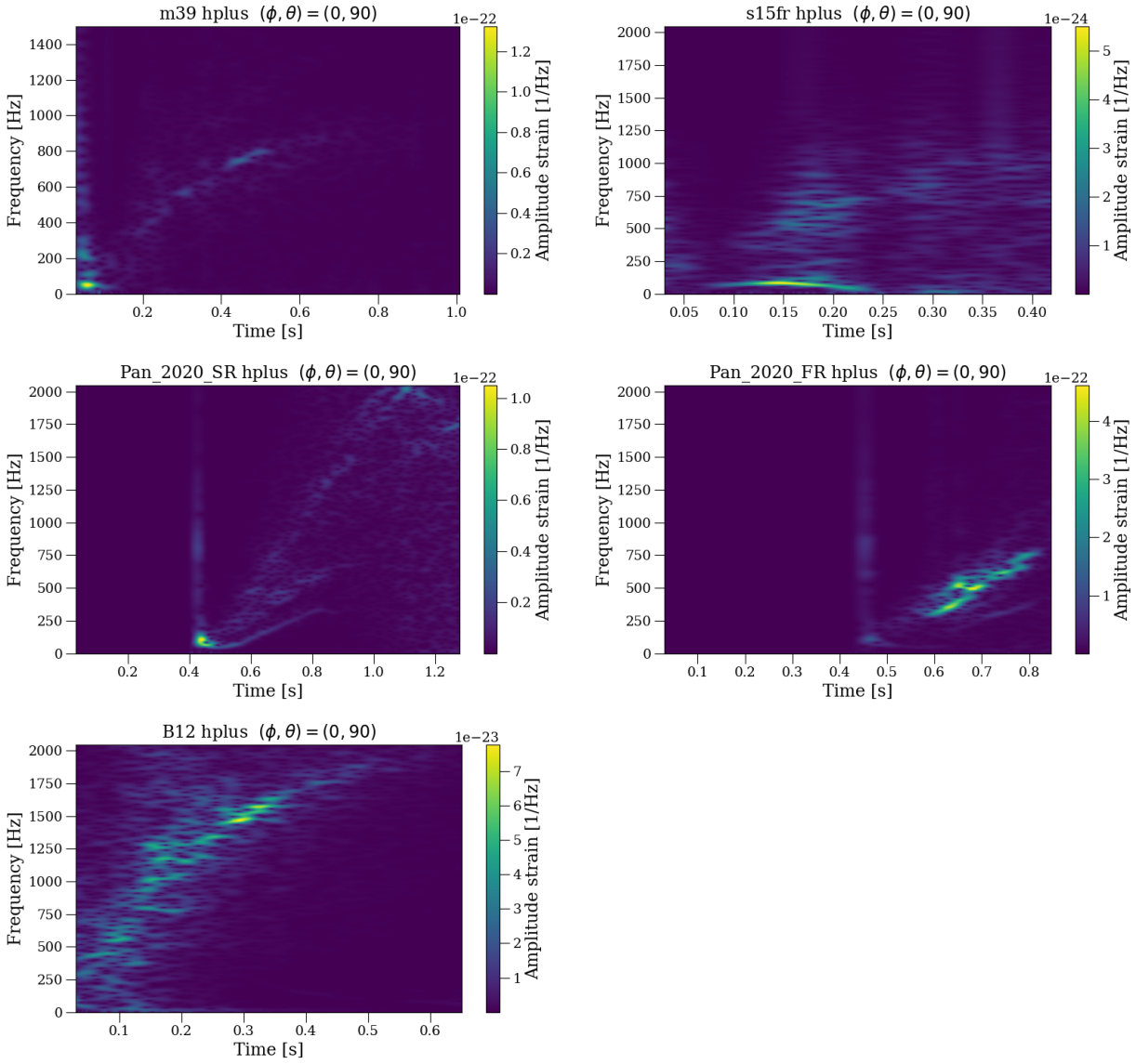

Figure 2.2: Time frequency representation of magneto-rotational model waveforms. The gravitational wave amplitude corresponds to a source distance of 10kpc.

Chapter 3

Introduction to deep learning

3.1 Deep learning fundamentals

While the emergent field of machine learning has become important in everyday life, this term is often misunderstood. Machine learning is the science of algorithms being trained on data and learning to perform a specific task without being explicitly programmed to complete it. Many outreaches, from autonomous car design [25] to the powering of clean energy [26] or the elaboration of advanced medicine techniques [27], are attributed to a branch of machine learning, named deep learning. Deep learning techniques have accomplished tasks that were thought impossible for machines while outperforming human capabilities. It is achieved through artificial neural networks, an ensemble of artificial neurons, designed to mimic the logical operations of the human brain.

3.1.1 Artificial neural networks

The starting idea of artificial neural networks, also simply called Neural Networks (NN), is to learn a given task from input data following a methodology inspired by brain information processing. Scientists started by designing the first unit composing biological brains, *i.e.* neurons.
The first mathematical model has been proposed by McCulloch and Pitts in 1943 [28]. They modeled a neuron via the Threshold Logic Unit with boolean inputs and outputs. Rosenblatt introduced the perceptron in 1957 [29], which generalizes the McCulloch and Pitts' neuron model for real inputs

$$\begin{aligned} f(\mathbf{x}) &= \begin{cases} 1 & \text{if } \sum_i w_i x_i + b \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ &= sign\left(\sum_i w_i x_i + b\right). \end{aligned} \tag{3.1}$$

The modern artificial neuron model enhanced the perceptron model by adding an activation function. It corresponds to a non-linear function. The model definition becomes

$$f(\mathbf{x}) = \sigma\left(\sum_i w_i x_i + b\right) \quad = \sigma\left(\mathbf{x}\mathbf{w}^\top + b\right), \tag{3.2}$$

which consists of the sum of the inputs x_i multiplied by the weight w_i, a bias b is added, and an activation function (here the sigmoid function σ) is applied to obtain the unit output $f(\mathbf{x})$. The artificial neuron represents the building blocks of neural networks. In the simplest neural network architecture, the multi-layer perceptron, units combined in parallel form a layer, and the composition of several layers form the neural network. The first layer and the last layer are respectively called the input layer and output layer. The layers between them are called the hidden layers. The layers could be fully connected or partially connected (some of the weights connecting two units are null). Figure 3.1 displays an illustration of a simple multi-layer perceptron. This simple NN, along with more complex networks, can learn meaningful information from data, supposing they are adequately trained.

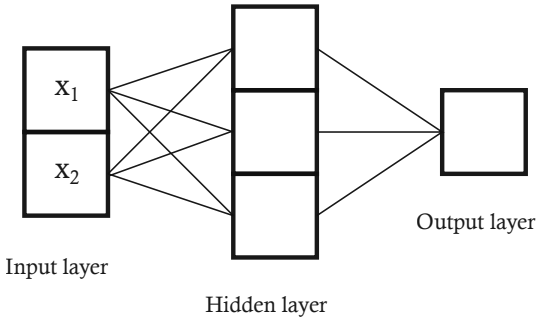

Figure 3.1: Illustration of a fully connected multi-layer perceptron with 2 inputs, a single hidden layer, and 1 output layer. Squares and solid lines represent respectively artificial neurons and connections between them.

3.1.2 Neural network training principle

A major aspect of neural networks is the learning phase, also called the training phase, after which the objective is to predict accurately desired outputs based on unseen data. During that stage, the network will independently adapt its parameters based on a given dataset via a numerical optimization method called gradient descent.

Gradient descent

Gradient descent is an iterative process optimizing a defined loss function $\mathcal{L}$ with respect to the model parameters θ (*e.g.* the weights and biases). The loss quantifies the prediction accuracy, *i.e.* how well the network outputs match the ground truth values. Hence, the learning process is guided by the finding of the loss (local/global) minimum.

The procedure's first step is to apply a Taylor expansion around the initial parameter values θ_0 of $\mathcal{L}(\theta)$ given a small perturbation ϵ with a penalty term

$$\hat{\mathcal{L}}(\epsilon;\theta_0) = \mathcal{L}(\theta_0) + \epsilon^T \nabla_\theta \mathcal{L}(\theta_0) + \frac{1}{2\gamma}\|\epsilon\|^2. \tag{3.3}$$

This amounts to approximating the loss function locally by a parabola (as illustrated in Figure 3.2 with the blue point representing θ_0). The $\hat{\mathcal{L}}(\epsilon;\theta_0)$ minimum is given by

$$\begin{aligned} \nabla_\epsilon \hat{\mathcal{L}}(\epsilon;\theta_0) &= 0 \\ \nabla_\theta \mathcal{L}(\theta_0) + \frac{1}{\gamma}\epsilon &= 0, \end{aligned} \tag{3.4}$$

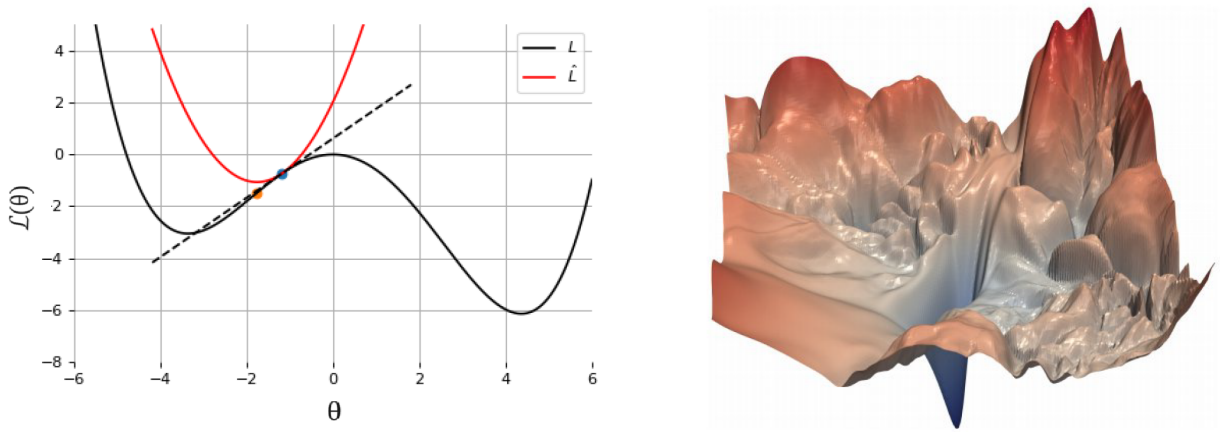

Figure 3.2: Left panel: second-order Taylor expansion around the initial parameter values θ_0 of $\mathcal{L}(\theta)$ given a small perturbation [30]. The blue point represents the parameter value θ_t at a given step t and the orange point highlights the loss value corresponding to the parabola minimum. Right panel: Loss surface of the deep neural network ResNet56 defined in [31]. The surface height represents the loss value in the projected space.

The parameter value at the next step θ_1 (in orange in Fig. 3.2) will be computed with

$$\epsilon = -\gamma \nabla_\theta \mathcal{L}(\theta_0). \tag{3.5}$$

Therefore, the updating rule is found to be

$$\theta_{t+1} = \theta_t - \gamma \nabla_\theta \mathcal{L}(\theta_t), \tag{3.6}$$

where θ_t is the value of the model parameters at time t and γ is a hyper-parameter called the learning rate. By repeating successively the preceding operation, the approximated loss will converge toward a minimum of the true loss. The key is now to be able to compute the loss gradient. As an illustration, imagine you are on the top of a hill and want to reach the valley. You decide in which direction to walk based on your current position and the slope of the mountain. Different methods exist to evaluate your next step, meaning to go down the hill.

In batch gradient descent, the gradient $\nabla_\theta \mathcal{L}(\theta_t)$ is computed by averaging the gradient of the loss $\nabla_\theta l(y_i, f(\mathbf{x}_i;\theta_t))$ between the n^{th} sample $\mathbf{x}_i$ and its ground truth value y_i over the whole dataset. The computational cost of an update step is proportional to the size of the dataset N. Nonetheless, the number of elements required to train deep learning models is at least on the order of thousands. One step becomes overly heavy to compute. Conversely, stochastic gradient descent is independent of N, as it randomly selects

a single dataset element at each iteration to calculate $\nabla_\theta \mathcal{L}(\theta_t)$ and update the parameter values. It makes it a computationally cheap process. It also becomes a stochastic process, resulting in a loss of some accuracy percentages due to the randomness of each step taken. However, network training does not require carrying out the minimization with high accuracy. Indeed, the objective is to perform a specific task accurately based on unseen data. The model must learn generic input patterns during training to be able to apply its knowledge to an independent dataset. While batch gradient descent performs the best in the minimization problem, stochastic gradient descent presents better generalization performances than batch methods given a computational budget [32]. The compromise between the two optimization algorithms is called mini-batch gradient descent. The total loss is determined over a dataset mini-batch *i.e.* a subset of samples. The update rule writes

$$\begin{aligned} \theta_{t+1} &= \theta_t - \gamma \nabla_\theta \mathcal{L}, \\ \nabla_\theta \mathcal{L} &= \frac{1}{B} \sum_{b=1}^{B} \nabla_\theta l(y_{n(t,b)}, f(\mathbf{x}_{n(t,b)}; \theta_t)), \end{aligned} \tag{3.7}$$

where $n(t, b)$ describes the order in which the samples are considered, either sequential or random. The size of the subset B is known as the (mini-)batch size. The method enables a balance between accuracy and convergence speed/duration depending on the mini-batch size chosen.

Backpropagation

The next question to tackle is the evaluation of the gradient $\nabla_\theta l(\theta)$ containing the partial derivatives of the loss with respect to each model parameter θ_k for $k = 0, ..., K-1$, K being the number of parameters. Let's consider a basic neural network, such as a multi-layer perceptron. Let's note $\mathbf{x}^l$ the output vector of the layer l, such that $\mathbf{x}^l = f(\mathbf{z}^l)$, generalizing the expression (3.2) for a neuron constituting a neural network. The $\mathbf{z}^l$ describes the input vector of the activation function with $\mathbf{z}^l = \mathbf{x}^{l-1}\mathbf{W}^{l\top} + \mathbf{b}^l$, with $\mathbf{W}^l$ being the weight matrix connecting the l^{th} layer neurons with the neurons of the layer $l-1$. Supposing a single output network and a loss function $(y - \hat{y})^2$ between the true output y and the NN prediction $\hat{y} = x^l$, the expression of its partial derivative with respect to some parameter w_*

$$\frac{\partial l(y, \hat{y})}{\partial \mathbf{W}_*} = \frac{\partial l}{\partial \mathbf{x}^l} \frac{\partial \mathbf{x}^l}{\partial \mathbf{z}^l} \frac{\partial \mathbf{z}^l}{\partial \mathbf{w}_*} \tag{3.8}$$

By applying the chain rule repeatedly, the third term can be determined

$$\begin{aligned} \frac{\partial \mathbf{z}^l}{\partial w_*} &= \frac{\partial \mathbf{z}^l}{\partial \mathbf{x}^{l-1}} \frac{\partial \mathbf{x}^{l-1}}{\partial \mathbf{z}^{l-1}} \frac{\partial \mathbf{z}^{l-1}}{\partial w_*} \\ &= \frac{\partial (\mathbf{x}^{l-1}\mathbf{W}^{l\top} + \mathbf{b}^l)}{\partial \mathbf{x}^{l-1}} \frac{\partial f(\mathbf{z}^{l-1})}{\partial \mathbf{z}^{l-1}} \frac{\partial \mathbf{z}^{l-1}}{\partial w_*}. \end{aligned} \tag{3.9}$$

The calculation is carried out recursively by going down the network until the derivative of the function directly depending on the parameter w_* is computed. Equations (3.8) & (3.9) mean that the total derivative of the loss is computable via a backward pass through the network layers. This procedure, called back-propagation, is feasible because neural networks are composed of differentiable functions. A forward pass is firstly realized to determine all the intermediate values (the $\mathbf{x}^l$, $\mathbf{z}^l$ values along with the output value) given

the inputs and parameters (weights and biases) values. Then, the partial derivatives are calculated via a backward pass. Equation (3.9) can be specified for a neuron j in a layer l

$$\frac{\partial z_j^l}{\partial w_*} = \sum_k w_{jk}^{l,l-1} \, f'(z_k^{l-1}) \, \frac{\partial z_k^{l-1}}{\partial w_*}. \tag{3.10}$$

where k index refers to a neuron k in the layer $l-1$. If the parameter w_* corresponds to a weight $w_{jk}^{l_*,l_*-1}$, the derivative of $z_j^{l_*}$ with respect to w_* is equal to the previous layer output $x_k^{l_*-1}$ and if the parameter is a bias $b_j^{l_*}$, the derivative will be equal to one.

For instance, let's reproduce the derivation for the 1-hidden layer MLP introduced previously. The derivative with respect to w_{11}^{10} (with the index $l=0$ describing the input layer, and the index $l=2$ the output layer) is

$$\frac{\partial l}{\partial w_{11}^{10}} = \frac{\partial l}{\partial x_1^2} \, f'(z_1^2) \, \frac{\partial z_1^2}{\partial w_{11}^{10}}, \qquad \frac{\partial z_1^2}{\partial w_{11}^{10}} = \sum_{n=1}^{3} w_{1i}^{21} \, f'(z_i^1) \, \frac{\partial z_i^1}{\partial w_{11}^{10}} = w_{11}^{21} \, f'(z_1^1) \, \frac{\partial z_1^1}{\partial w_{11}^{10}},$$

which leads to

$$\frac{\partial l}{\partial w_{11}^{10}} = \frac{\partial l}{\partial x_1^2} \, f'(z_1^2) \, w_{11}^{21} \, f'(z_1^1) \, x_1^0.$$

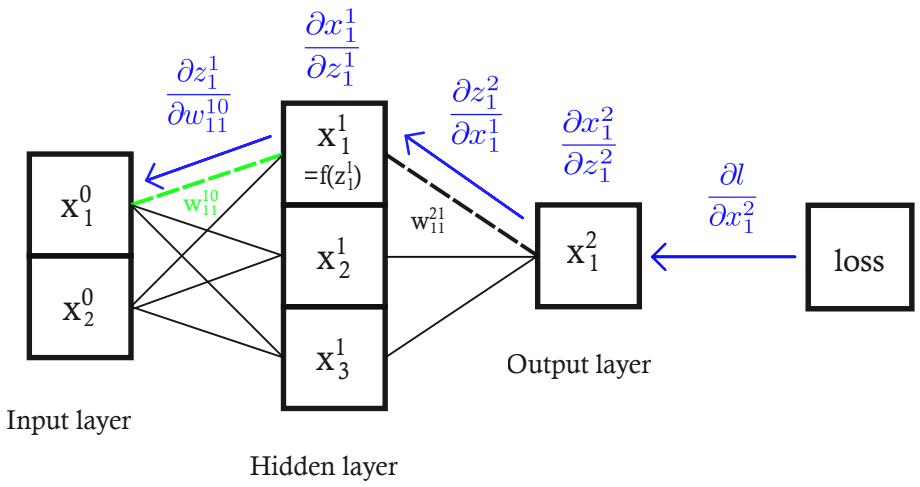

Figure 3.3: Illustration of the backpropagation procedure in a fully-connected multi-layer perceptron. The blue arrows indicate the backward pass path and the partial derivatives computed along it. The parameter highlighted in green corresponds to the parameter with respect to which the loss derivative is calculated.

The backward pass path is represented in Figure 3.3. The successive computations of the partial derivatives are displayed in blue. It clearly shows that each partial derivative corresponds to a backward step through the network until it encounters the parameter w_* being here w_{11}^{10}. Note that the forward pass is needed beforehand to be able to run the backward pass.

3.1.3 Neural network training ingredients

The gradient descent along with backpropagation is an effective method for simple and small neural network training. However, the training of deep networks need additional ingredients.

Activation function

Neural networks are composed of artificial neurons. Their implementation involves an activation function. They have been added to the Rosenblatt model to introduce non-linearity in the model. Indeed, Minsky and Papert have shown in their book *Perceptrons* [33] that Rosenblatt's model (3.1) performs efficiently only on linearly separable patterns and shows bad competence in learning generalization based on learned examples.

The sigmoid function is part of the first activation functions introduced to build deep neural networks. However, network training was challenging due to the problem of vanishing gradient. The vanishing gradient problem occurs when the NN is composed of many layers. The succession of partial derivatives in the gradient computation corresponds to a multiplication of small numbers by small numbers leading to a small gradient value. In that case, gradient descent is slowed down or even stopped resulting in a limited learning capacity. It is also true when using hyperbolic tangent or more generally bounded activation functions. A solution was proposed by [34], introducing the ReLU (Rectified Linear Unit) activation function. ReLU, and its variation functions, such as Leaky ReLU [35] and ELU [36] composed modern NN. They are represented in Figure 3.4 along with some other existing activation functions.

In addition, note that activation functions must also be differentiable (enabling backpropagation as stated in the previous section) and must have a simple mathematical expression in order to complete the training procedure in a reasonable amount of time.

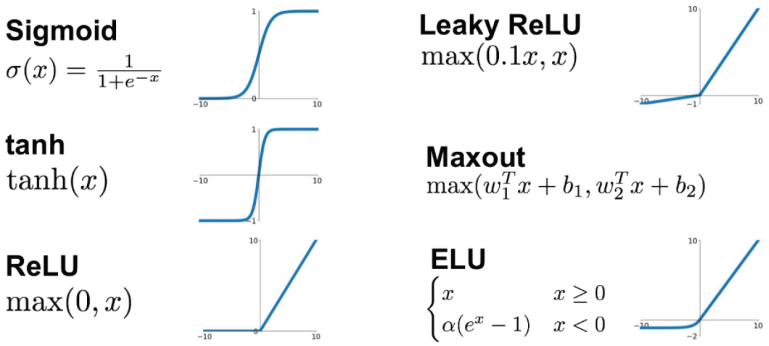

Figure 3.4: Examples of activation functions with their mathematical expression and respective plot [6].

Optimizers

In general, deep neural network training is long and complex. In the basic setting of gradient descent over mini-batches, the algorithm presents limitations. The gradient is optimized via several techniques.

One likely situation corresponds to gradient descent encountering a flat region in the loss landscape, *i.e.* a region with consistently small gradients. The optimization algorithm will move slowly and might not converge within a reasonable time. Intuitively, the addition of inertia would enable the algorithm to pass local barriers and accelerate the computation when the gradient is not changing considerably, similar to a ball rolling down a hill. Momentum is introduced in the update rule (3.7) influencing the step direction choice. It is implemented as follows

$$\begin{aligned} u_t &= \alpha u_{t-1} - \gamma \nabla_\theta \mathcal{L}, \\ \theta_{t+1} &= \theta_t + u_t. \end{aligned} \tag{3.11}$$

The variable u_t encodes the velocity namely, the direction and the speed of the parameter evolution at the time step t. The hyper-parameter α will balance the influence of the recent gradients on the next velocity computation and on gradient descent evolution.

An assumption made via the use of vanilla gradient descent is the curvature isotropy such that the learning rate parameter (characterizing the update step size) is a constant set before the network learning phase. It means that each parameter is updated uniformly regardless of the curvature. However, an anisotropic situation might result in an inefficient optimization. Algorithms using adaptive learning rates, such as AdaGrad [37] and RMSProp [38], include a per-parameter adaptation of the learning rate. The optimizer used as default is named Adam [39]. Adam combines momentum, accelerating gradient descent, and RMSProp, implementing a learning rate downscale depending on historical gradient values (with more recent values weighing more than older ones).

Initialization

In the context of gradient descent, minimizing a loss is needed to be able to apply our parameter update rule and find the optimal model parameter set. Given suitable learning rate value and activation functions combined with optimization techniques described in the previous section, the optimization problem convergence is well assured (regardless of the initialization conditions). However, this is true for convex problems. In a non-convex regime, the loss expression might be complex, potentially entailing several local minima, saddle points, flat regions, and varying curvatures. Hence, the choice of parameter value initialization is important. For example, Figure 3.2 illustrates the complex loss surface of the deep neural network ResNet56 [40]. Two main strategies implemented in modern NN are the Xavier initialization [41] and the He initialization [42]. These have been developed based on two conditions: breaking the symmetry and controlling the variance and the scale of the weights. The second condition expresses that the weight variance must be preserved during the forward and backward pass *i.e.*, ensuring the information keeps flowing through the layers without the signal magnitude vanishing or exploding.
The weights are drawn from a distribution with a variance

$$\begin{aligned} \textit{Xavier initialization :} \quad & \mathbb{V}[w^l] = \frac{2}{q_{l-1} + q_l}, \\ \textit{He initialization :} \quad & \mathbb{V}[w^l] = \frac{2}{q_{l-1}}, \end{aligned} \tag{3.12}$$

with l the layer index and q_l the width of the l^{th} layer.

Normalization

The strategies developed in the previous section make the hypothesis of input features with equal variance. This is not always true and can be enforced by data normalization. This is accomplished by standardizing the input data feature-wise. The resulted input data features have a zero mean and a unitary variance. This is realized by subtracting from the feature vectors their mean value and dividing by their standard deviation value.

An additional method managing the variance across layers is named batch normalization. The change in layer parameters directly impacts the input of the next layer, resulting in a change in the distribution of each layer's inputs during the training phase. Batch normalization [43] reduces this phenomenon, facilitating the NN training. The normalization is integrated into the model architecture. It is applied between the NN layers with the mean and variance values computed over each mini-batch. Given of mini-batch $\mathcal{B} = \{\mathbf{x}_1, ..., \mathbf{x}_B\}$, the batch normalization transform formulae are

$$\hat{\mu}_{\mathcal{B}} = \frac{1}{B}\sum_{n=1}^{B}\mathbf{x}_i, \qquad \hat{\sigma}^2_{\mathcal{B}} = \frac{1}{B}\sum_{n=1}^{B}(\mathbf{x}_i - \hat{\mu}_{\mathcal{B}})^2, \qquad \mathbf{x}'_i = \gamma\frac{\mathbf{x}_i - \hat{\mu}_{\mathcal{B}}}{\sqrt{\hat{\sigma}^2_{\mathcal{B}} + \epsilon}} + \beta, \tag{3.13}$$

with $\hat{\mu}_{\mathcal{B}}$ and $\hat{\sigma}^2_{\mathcal{B}}$ the batch mean and variance and $\mathbf{x}'_i$ the transformed layer input value. The parameter ϵ is a constant introduced for numerical stability, and the parameters γ and β are learned by the neural network. The technique speeds up the training, reducing the number of steps needed to reach a given accuracy compared to data normalization [43].

3.1.4 Neural network training

The training phase of an artificial neural network is a central concept behind their prediction capacities. It is part of a specific procedure for the algorithm to perform accurately.

Procedure

Several learning contexts exist in deep learning, including supervised learning. The algorithm will learn from labeled samples, meaning each dataset sample possesses a ground truth value. Ground truth values, also called the target values, are the desired output for the algorithm to predict. This context is suitable for the learning process of artificial neural networks. Indeed, it involves the minimization of a loss function, which compares the network prediction with the target values. The loss function represents the problem we want to solve, either a classification or a regression problem, and the application we are considering. The different kinds of losses will be detailed in the following section.

The output of the learning algorithm will be a model learned on a dataset. The algorithm defined with a number of parameters (*e.g.* the number of layers, the number of neurons in layers, or the number of training epochs) aims at generating a model predicting with high accuracy the target values. Nonetheless,

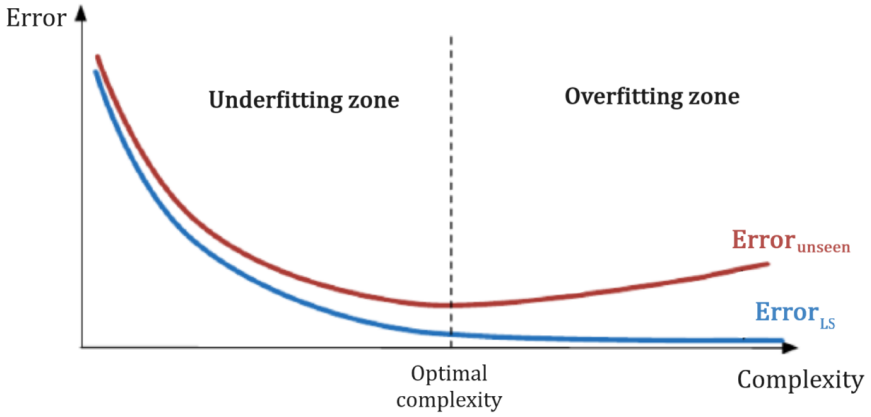

Figure 3.5: Adapted from [44]. Illustration of the evolution of the training error in blue called $Error_{LS}$, and the test error in red named $Error_{unseen}$, as a function of the model complexity. The dotted line represents the optimal model complexity.

the model may be too simplistic, in that case, the algorithm is said to underfit, or it may be too complex and the algorithm overfits. The former scenario happens when the algorithm is incapable of learning meaningful information and patterns from input data. In the latter situation, the algorithm has learned the input characteristics too specifically, without being able to generalize the learned information. More precisely, the algorithm will overfit when it begins to fit the data noise. Both cases lead to poor performance.

As a consequence, the training error, *i.e.* the error computed on the training set, is not representative of the error made on unseen data. The solution is to evaluate the model performance on an independent dataset from the dataset called the test set. The behavior of the training and test error as a function of the model complexity is represented in the figure 3.5. When the model overfits, the error on unseen data (the test error) will increase, while the training error decreases. Conversely, the model underfits when the training and test errors increase with a decreasing complexity. The optimal complexity is selected such as the test error is minimal.

In practice, a validation set is introduced to perform model selection and fine-tune hyperparameters. The final performance is evaluated on the test set. The most important hyperparameter to focus on is the learning rate. It will directly influence the learning capacities of the learning algorithm. Indeed, since it directly impacts the step size taken after each iteration, a large value means a chaotic gradient descent path and potentially missing the loss minimum. Whereas too small values would slow down or stop gradient descent. In both cases, it results in limited learning capacities and poor results.

Loss function

In deep learning, the loss function is a scalar function selected depending on the application considered. For instance, the cross-entropy loss is a common loss function for classification problems. For a multi-class

classification, the categorical cross-entropy loss function for a given sample n is defined by

$$l_n = -\sum_{c=1}^{C} y_{n,c} \log p_{n,c}, \tag{3.14}$$

with C the number of classes, p_c the network prediction probability for the element n belonging to the class c and $\mathbf{y_n}$ a vector containing a single one value at the ground truth class index and containing zero values otherwise. The probabilities are computed through the Softmax function applied on the input tensor $\mathbf{x}_n$

$$Softmax(x_{n,c}) = \frac{\exp(x_{n,c})}{\sum_{c'=1}^{C} \exp(x_{n,c'})}, \tag{3.15}$$

rescaling the input tensor element values in the range [0,1] with their sum along a given dimension equal to one. A basic loss function for a regression problem is the squared loss function defined by

$$l_n = (\mathbf{y}_n - \hat{\mathbf{y}}_n)^2, \tag{3.16}$$

taking the squared difference between the ground truth value y_n and the network prediction $\hat{y}_n$.

3.2 Convolutional neural networks

Convolutional Neural Networks (CNN) are neural networks suitable for processing images in computer vision tasks. This stems from CNN's intrinsic properties namely, translation invariance and equivariance, local capture of spatial features, and a hierarchical feature composition. CNNs comprise convolutional layers, pooling layers, and fully connected layers with ReLU activation functions.

3.2.1 Convolutional layer

The convolution operation between two tensors is simple. A tensor, called a kernel, with a lower size than the input tensor slides over the input tensor. The input elements encompassed in kernel projection onto the input are multiplied by the superimposed kernel element. The principle of a 1D convolution operation is illustrated in Figure 3.6.

A convolutional layer is defined by K kernels u_k of sizes $C \times h \times w$ and applies a 2D discrete convolution operation[1] on an input tensor $\mathbf{x}$ of a size $C \times H \times W$ to generate K output feature map o_k of size $(H - h + 1) \times (W - w + 1)$. The output element ij is determined by the expression

$$[u \circledast \mathbf{x}]_{i,j} = \sum_{c}^{C-1} \sum_{m}^{h-1} \sum_{n}^{w-1} u_{c,i+m,j+n}\, x_{c,m,n}. \tag{3.17}$$

The C letter denotes the number of channels of the input tensor and the kernels. Note that the latter expression contains a dot product, interpreted as a similarity measure between the tensors. By displaying kernels in a chosen layer, patterns learned from the input could be observed *e.g.* horizontal, vertical lines,

[1] Strictly speaking, this operation is not the common convolution operation since both tensor elements are considered with indices in ascending order

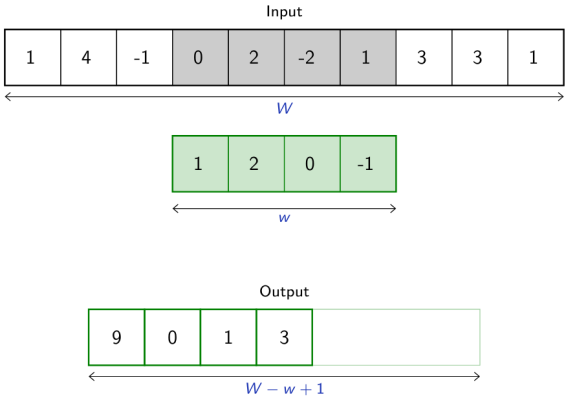

Figure 3.6: Illustration of 1D discrete convolution operation between an input vector of size W and a kernel of size w. The output vector has a size $W - w + 1$ [45].

or circular arcs. These layers possess additional hyperparameters: the padding, the stride, and the dilation parameter. The stride parameter corresponds to the step taken by the kernel to slide over the input tensor. The stride parameter effect is represented in Figure 3.7 for a stride set to one. It allows us to control the reduction of the input spatial dimension by a constant factor. The padding corresponds to the frame size added around the input tensor (see figure 3.8 for an illustration of a unitary padding). It enables controlling the output map size, *e.g.*, preserving the input size across layers. The dilation parameters determine the size of zeroed columns and rows inserted between the kernel elements. Figure 3.9 displays the situation of a dilation value equal to one. For non-zero dilation value, the kernel, along with its receptive field size, is expanded. The receptive field of an output component is the sub-area in the input image influencing it indirectly through previous layers.

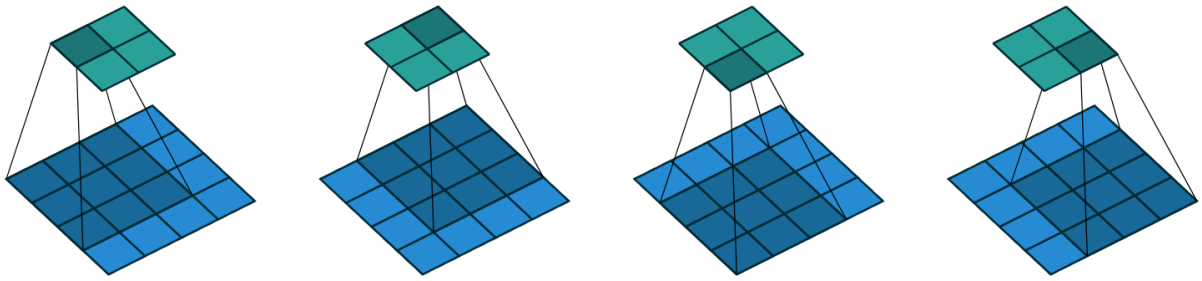

Figure 3.7: Convolution operation between 3×3 kernel and a 4×4 input tensor with a stride parameter equal to 1 with no padding and no dilation [46].

A parallel with fully connected layers can be drawn to deepen our insight. Convolution layers are specific FC layers, with local and shared weights as represented in Figure 3.10 b) situation. Each weight value linking neurons involved in the convolution operation is a kernel element value. The other weights are null. The locality property stems from the local connections of neurons, resulting in defined receptive field. The Figure 3.10 highlights that weight intensities are shared across neurons in convolution layers. The translation

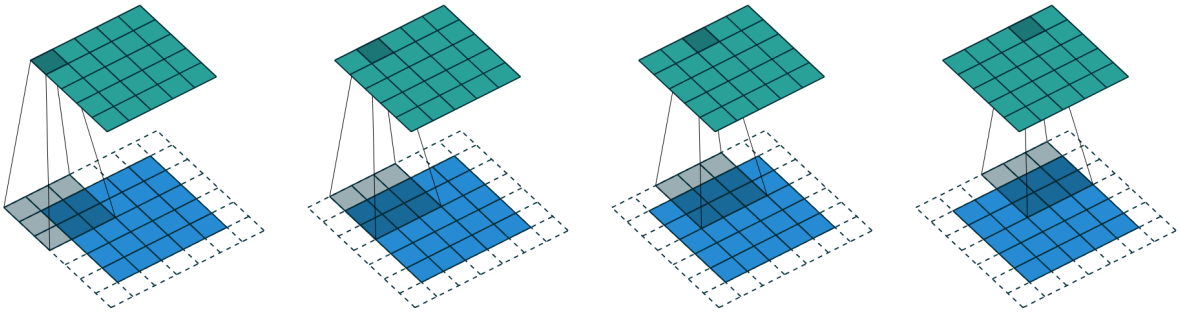

Figure 3.8: Convolution operation between 3×3 kernel and a 5×5 input tensor with a padding parameter equal to 1 with the stride set to one and zero dilation [46].

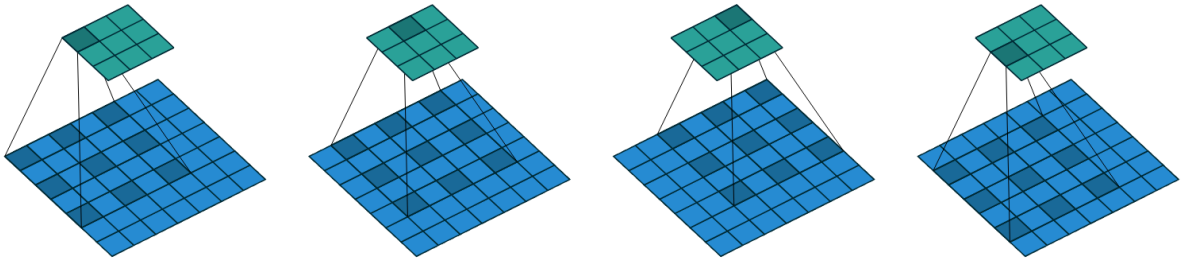

Figure 3.9: Convolution operation between 3×3 kernel and a 7×7 input tensor with a dilation parameter equal to 1 with the stride set to one and zero padding [46].

equivariance property is inherited from the latter fact. Note that the weights are learned in FC layers whereas the kernel elements are the learnable parameters in convolutional layers.

3.2.2 Pooling layer

Pooling layers reduce the input spatial dimension allowing the network to capture more global features. Pooling operations preserve the input structure via either max-pooling or average pooling. Considering a pooling area $h \times w$, an output tensor o of size $C \times r \times s$ is computed such as

$$\begin{aligned} Max - pooling : o_{c,i,j} &= \max_{m<h, n<w} x_{c,\, ir+m,\, js+n}, \\ Average\ pooling : o_{c,i,j} &= \frac{1}{hw} \sum_{m}^{h-1} \sum_{n}^{w-1} x_{c,\, ir+m,\, js+n}, \end{aligned} \tag{3.18}$$

from input tensor of size $C \times sh \times rw$. That is to say, the 3D input tensor is divided into several 2D sub-tensors, and the average or the maximum value of each is computed to generate the 3D output tensor. The pooling operation provides a pseudo translation invariance to CNNs. Local changes in the pooled areas will not impact the resulting output tensor.

A common CNN structure comprises M successive blocks $[[Conv \rightarrow ReLU] * N \rightarrow Pooling]$ followed by a block $[[FC \rightarrow ReLU] * K \rightarrow FC]$ with $N \leq 3$, $M > 0$, $K < 3$. Pooling layers ($Pooling$) (or

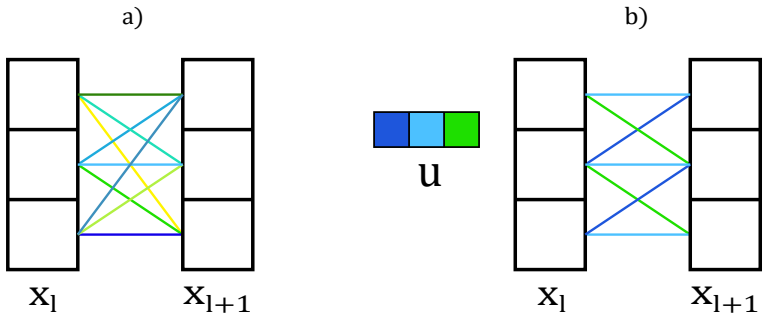

Figure 3.10: The two situation represents two network layers $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$. Squares and solid lines represent respectively artificial neurons and connections between them. u is a 1×3 kernel. The weight and kernel element values are characterized by a specific color. The a) and b) cases illustrate fully connected and convolution layers.

strided convolutional layers) and convolutional layers (*Conv*) are alternate, leveraging the spatial dimension reduction of input tensors. Considering kernels at increasing network depths, the input sub-area captured by them increases. It allows the network to learn global features. Moreover, convolutional operations are stacked. It increases the effective receptive field size of a given kernel while introducing less parameters. For instance, two stacked 3×3 kernels have an equal effective receptive field size as a single 5×5 kernel, with fewer parameters to learn (18 against 25 for a kernel size 5×5).

Chapter 4

Anomaly detection for Short duration BUrst Searches

ALBUS, stating for Anomaly detection for Long duration BUrst Searches, is a convolutional neural network designed for the search of long-duration bursts by Vincent Boudart [6]. It is commonly known that CNNs perform well on images for computer vision tasks. One could use them to perform gravitational wave detection based on spectrograms. For instance image classification tasks in gravitational wave searches have presented great results, for example with Time-Frequency (TF) representation classification of compact binary coalescent [47] or glitches [48]. In addition, deep learning techniques are interesting for their rapidity in input processing. Hence, the algorithm could carry out low-latency searches, producing a GW signal detection alert within seconds and facilitating follow-up searches. Observatories would be alerted enabling the monitoring of the detected event's electromagnetic and/or neutrino counterparts.

CNNs classify spectrograms as part of the signal category based on a sole probability value. If not considered as containing a signal, the spectrogram is classified into the noise category. Nonetheless, detecting a gravitational wave signal in the interferometer noise remains a challenging task. Many signals may have an intensity just above the detector sensitivity and hardly emerge from the noise in spectrograms. Relying on a single value could be risky and signal detection is likely to be missed. In those circumstances, ALBUS has been designed as a non-linear noise filter, extracting from an input spectrogram the pixels corresponding to an identified candidate GW signal. Promising results have been obtained for long-duration GW signals with ALBUS [6].

Deep learning detection techniques have also been considered for short-duration burst signal searches, especially for core-collapse supernova signal research. As part of the first studies in this context, Astone *et al.* [49] make use of a convolutional neural network to recognize the peculiarities in CCSN signals in spectrograms. The training signals correspond to phenomenological waveforms, which are injected into colored Gaussian noise. Additionally, Iess *et al.* began by investigating a 1D and 2D CNN [50] before going further in their analysis and include the training of a recurrent neural network on simulated waveforms [51]. Chan *et al.* [52] investigated a multi-class classification problem. They trained a CNN to identify neutrino-driven and magneto-rotational waveforms embedded in Gaussian noise with spectral properties of the LIGO and

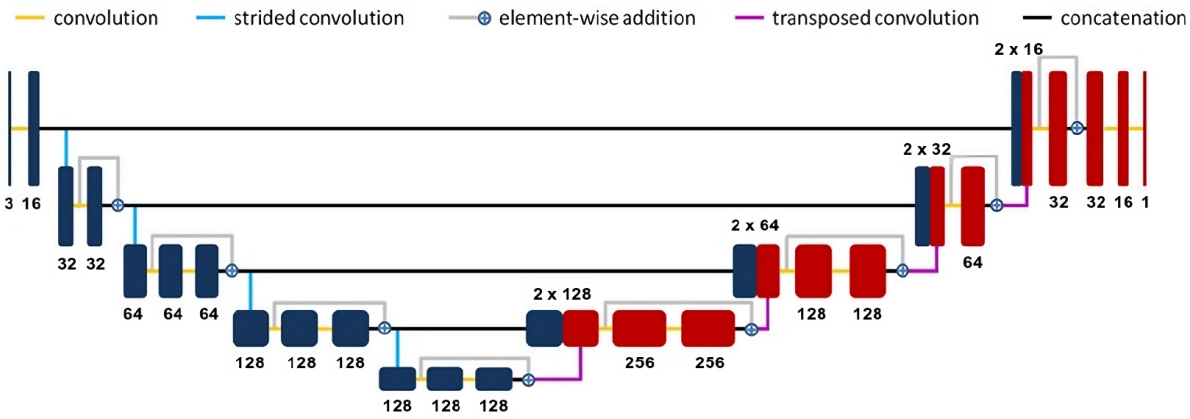

Figure 4.1: Illustration of the UNet architecture [6], which comprises two sections: a down-sampling part in blue and an up-sampling part in red. They are both connected through skip connections drawn in black solid lines. The black numbers are the number of feature maps at each step in the CNN. The network output contains the extracted signal pixels.

Virgo detectors. López *et al.* [53] refined the phenomenological waveform treated in [49] and injected them into LIGO-Virgo detector noise. They utilized a CNN to classify the signal time-frequency representation. The interpretability of CNN models in CCSN signal detection has been tackled by Sasaoka et al.[54] by implementing class activation mapping, which provides visual information on the input regions utilized to generate the model's output.

These articles consider either a binary or a multi-category classification problem, thus relying on class probability for signal detection. Extending the intention behind ALBUS, our project aims to adapt ALBUS for short-duration signal detection, focusing on core-collapse supernova signals. Before going into details about the adaptations for short-duration signals, we will describe ALBUS architecture.

4.1 ALBUS architecture

ALBUS is a specific type of convolutional neural network called a UNet. The architecture is represented in Figure 4.1. It is composed of an encoder, down-sampling the input image, and a decoder, up-sampling back to the initial input spatial dimension. The output, named hereafter the anomaly map, contains the pixels corresponding to the identified candidate signal.

As expressed in Chapter 3, convolution layers associated with strided convolution layers extract more global spatial features in the image while reducing the feature maps resolution. As the information flows through the encoder layers, learned patterns are combined, generating increasingly intricate patterns. Starting with high-resolution feature maps, the decoder generates low-resolution feature maps containing general information.

The inverse operation is carried out through transposed convolution layers. These layers implement a transposed convolution operation, up-scaling an input tensor and are defined in the same manner as convolutional layers, with the same parameters. However, in transposed convolution, the stride and padding

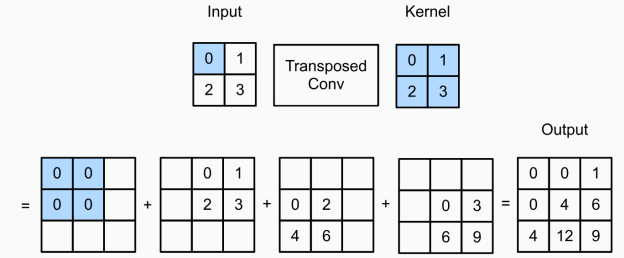

Figure 4.2: Illustration of the computation of a 2D transposed operation on a 2×2 input tensor with a 2×2 kernel, a stride value of one, and zero padding and dilation [55].

parameters are defined on the output tensor such that if the output is convolved, the input tensor dimension is retrieved. Figure 4.2 exemplifies the computation procedure of a transposed convolution output tensor in a 2D case.

The sole information contained in low-resolution maps may be insufficient for precisely recreating and localizing the features of interest. A solution implemented in the UNet is known as skip connections. Skip connections are concatenation operations linking UNet's parts at different resolution levels. Hence, the decoder layers benefit from low-resolution maps from the previous layer and detailed feature maps from the encoder. Overall, the encoder identifies the relevant information in the input, and the decoder samples them up to the input spatial dimension.

This architecture is not restrained to a specific input size and could be adapted for short-duration signals.

4.2 ASBUS - ALBUS adaptations

A few adaptations have been implemented in both the dataset and the architecture of ALBUS to perform short-duration burst detection.

4.2.1 Time-frequency representation

The first adaptation consisted of finding a suitable time-frequency representation, which must optimize the signal visibility. Single-detector spectrograms have proven to be appropriate. The spectrograms have been generated through the short-time Fourier transform technique. It divides the signal into overlapping segments and computes the Fourier transform of each segment. This technique is particularly suited for short and non-stationary signals. It was performed on three seconds of data sampled at 4096 Hz (leading to a spectrogram frequency range from 0 to 2048 Hz). Spectrograms commonly characterize the energy evolution as expressed in Section 1.4. In this work, however, they characterize the Fourier transform magnitude, *i.e.* the absolute value of the Fourier transform.

4.2.2 Target maps

Each dataset element must be associated with a target map in a supervised learning setting. A target map represents the ground truth image, *i.e.* what the neural network aims to predict. Their generation requires careful attention, as they are an important piece impacting the result's accuracy. Indeed, while the

network adapts its predictions based on the target maps, ALBUS outputs are at most accurate than the ground truth images.

Core-collapse supernova signals are widespread in spectrograms over time and frequencies (as seen in Fig. 2.1 and Fig. 2.2). Thus, the extraction of the whole signal with all its features is challenging. We decided to focus on isolating the signal core, namely, the loudest and most identifiable part of the signal. For CCSN, the most common feature is the arch-shaped pattern seen in spectrograms. In order to create the target maps, we implemented a procedure optimized for isolating the latter pattern.

The target map computation is a two-step procedure: The injection mask is generated first, followed by the target map. The injection mask is a boolean map situating the injected signal pixels in the spectrograms. For the same reason mentioned above, we will focus on isolating the injection core. The injection mask computation sub-steps are depicted in Figure 4.3. Let us develop it.

Since we inject the signal into the noise, we base the feature isolation on the subtraction of the spectrogram before the injection (which contains only noise) from the one with the injection. This way, the pixels affected by the injected signal are extracted. The resulting map is henceforth referred to as the difference map (corresponding to the second panel). Then, difference map 2% highest value pixels are saved as a boolean map (cf. third panel). Any pixel clusters resulting from noise artifacts are removed outside the signal duration. The removed areas are highlighted in yellow in the fourth panel. Note that these boolean map parts are set to zero in practice. The fifth, sixth, and seventh panels display morphological operations implemented in the python package *Scikit-image* [56]. The transformations are employed for the purpose of creating a filled mask. We applied successively, small object removal (of all clusters with an area under 1000 pixel^2), a feature expansion by 10 pixels, and a morphological closing on a distance of 50 pixels in the vertical and horizontal direction. Closing manipulation applies a dilation followed by its inverse operation, namely the erosion operation. Through this sequence, small dark regions tend to be removed and bright areas tend to be connected. Under the hypothesis that the highest area cluster is the injection cluster, the final sub-step entails the latter selection, thereby forming the definitive injection mask.

The target map is generated in a similar manner, the only difference being in the two first steps. The difference map is computed based on spectrograms where only the pixels superimposed with the injection mask are non-null. Then, the pixels with a value lower than the 99% quantile of the difference map are discarded. The injection mask and the resulting target map are shown in Figure 4.4. The final target incorporated the magnitude of pixels identified as belonging to the injected signal. Figure C.1 shows examples of target maps for three additional CCSN models.

4.2.3 Neural network architecture

ALBUS architecture takes as input spectrograms of a given resolution. Nonetheless, ALBUS' implementation does not handle variable-size input tensors. The architecture's hyperparameters have been changed to match the short-duration time-frequency representation resolution. The first layer is not the only layer to adapt as ALBUS includes concatenation operation joining the encoder and decoder. In other words, the feature map size at each resolution level must match to be concatenated. The formulae 4.1 gives the convolution and transposed convolution output size depending on the layer parameter values, such as the

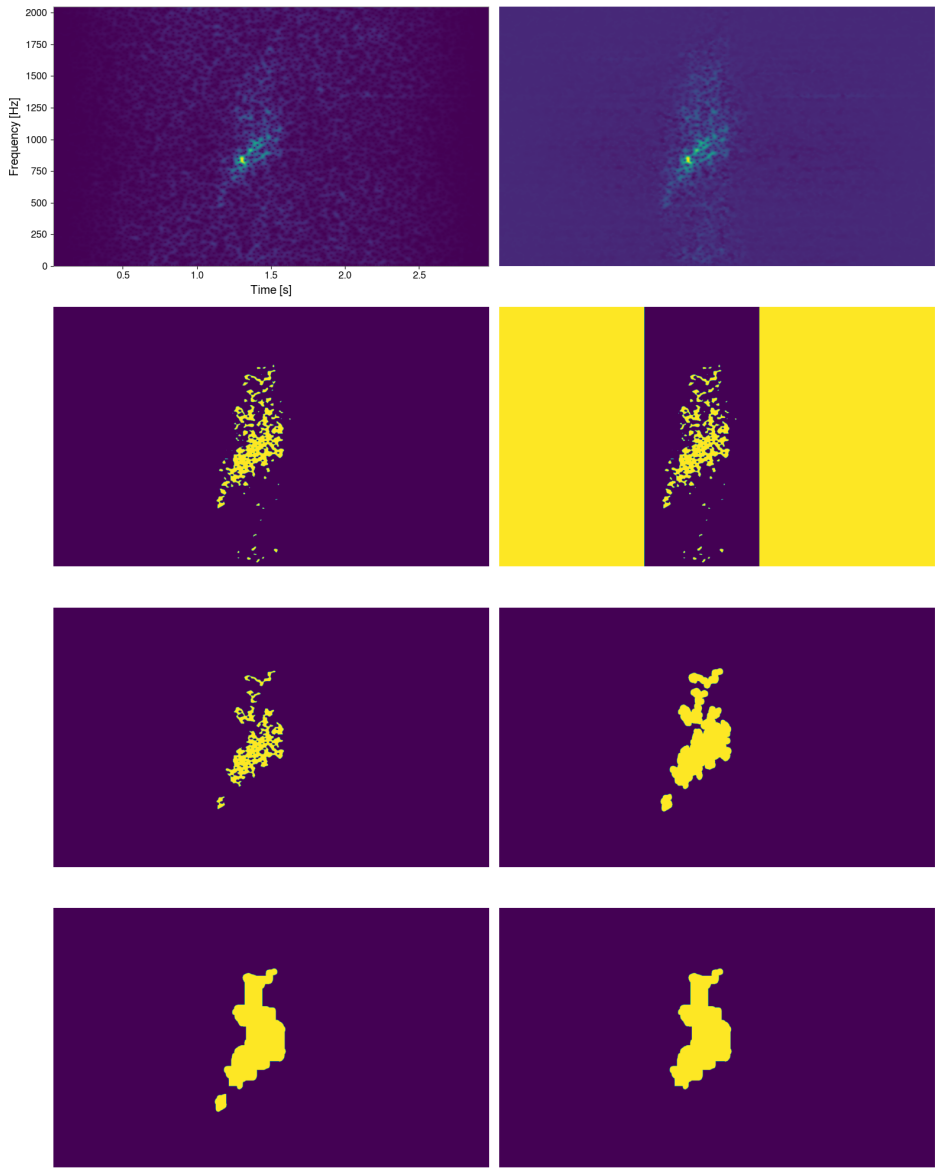

Figure 4.3: Injection mask computation procedure illustrated for a *s18* waveform spectrogram instance. The first panel depicts the waveform spectrogram. The other panels depict each step of the injection mask computation of the model considered. The last panel displays the final boolean mask. Note that for the sake of illustration, the boolean map area outside the injection duration in the fourth panel has been set to one instead of zero.

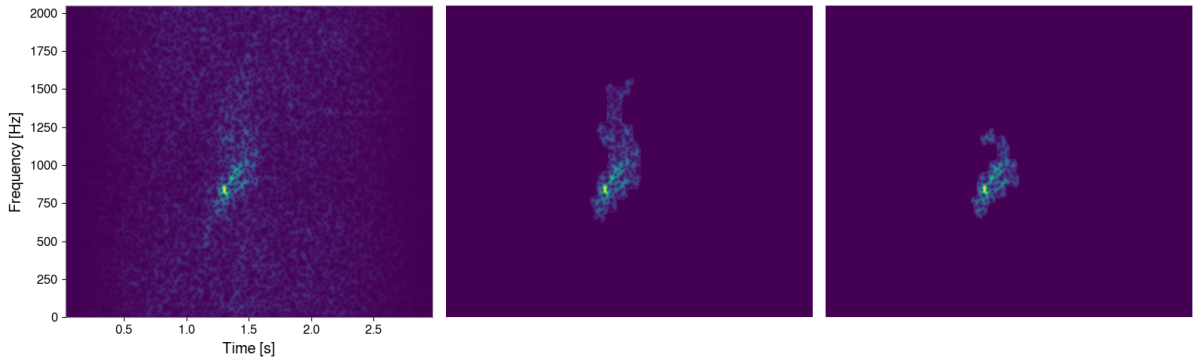

Figure 4.4: First panel: *s18* model spectrogram. Second panel: *s18* spectrogram pixels superimposed with the injection mask. Third panel: *s18* spectrogram target map.

stride (s), the padding (p), the dilation (d), and the kernel size (k). The input and output sizes are indicated by the letters i and o. op represents a padding value applied to the output. The two formulae are true for the output height and width. One can tune the layer hyperparameter values based on these equations to obtain the desired output feature map resolution. Table D.1 gathers ALBUS layer hyperparameter values suitable for our project purpose. The entire architecture is detailed in [6] Appendix A. Note that the first zero-padding layer is removed (22^e operation in Boudart table).

$$\begin{aligned} Convolution &: o = \frac{i + 2p - d(k-1) - 1}{s} + 1, \\ Transposed\ convolution &: o = (i-1)s - 2p + d(k-1) + 1 + op. \end{aligned} \tag{4.1}$$

Table 4.1
Hyperparameters of convolution-type operations in ALBUS architecture. The adaptations for short-duration signal searches are highlighted in bold.

Layer	Kernel size	Stride	Padding	Dilation	Output padding
Conv[a]	(3,3)	(1,1)	(1,1)	(1,1)	/
SConv[b]	(3,3)	(2,2)	(1,1)	(1,1)	/
TConv[c]	(3,3)	(2,2)	(1,1)	(1,1)	**(1,0)**

[a] Conv: Convolutional layer.
[b] SConv: Strided convolutional layer.
[c] TConv: Transposed convolutional layer. Out of four, the output padding is introduced in the first three TConv.

Finally, ALBUS became ASBUS, which stands for Anomaly detection for Short-duration BUrst Searches. The next important step corresponds to ASBUS training. For this purpose, we need to generate a dataset.

4.3 Dataset

The dataset is composed of two spectrogram classes: injection and background spectrograms.

The background category contains time-frequency representations of gravitational wave detector noise.

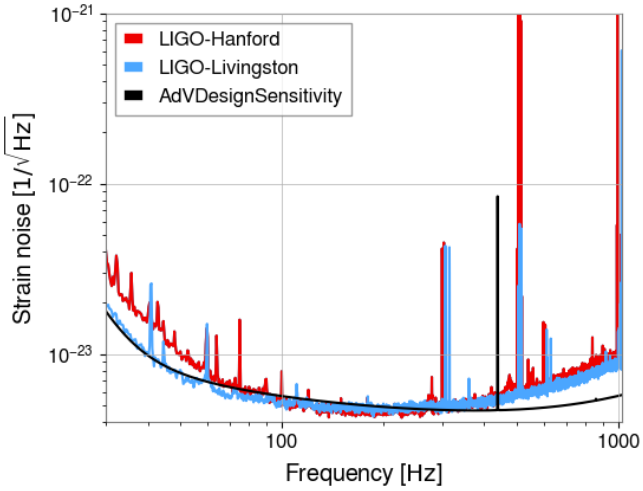

Figure 4.5: Advanced LIGO 2021 design sensitivity curve (named AdvDesignSensitivity) compared to sensitivity curves of Hanford and Livingston LIGO detectors during the O3 observing run.

Its generation is based on the PSD of an Advanced LIGO 2021 design sensitivity scenario from the package *LALSimulation* [57]. It corresponds to Gaussian noise with LIGO spectral features. Figure 4.5 compares the considered LIGO design sensitivity curve to the LIGO sensitivity curves during the third observing run, which lasted from April 2019 to March 2020.

The second category's spectrograms contain one core-collapse supernova waveform injected into noise. The waveform amplitude is calculated based on 3D simulated quadrupole moments following equation (1.20). However, a preprocessing is necessary beforehand. The simulations are resampled at 4096 Hz and scaled as emitted from a distance of 10 kpc. After resampling, a high pass filter is applied with a cutoff frequency of 11 Hz, removing resampling artifacts. The plus and cross polarizations are computed for uniformly sampled sky position (θ, ϕ). Subsequently, the simulation amplitude $h(t)$ is calculated by taking into account the Earth's rotation in equation (1.26)

$$h(t) = F_+(\theta, \phi, \psi, t)\, h_+(t + \Delta t) + F_\times(\theta, \phi, \psi, t)\, h_\times(t + \Delta t), \tag{4.2}$$

with Δt the time delay between a given detector and the assumed geocentric frame. In other words, h_+ and $h_\times$ projection onto the detector frame enables recovering the waveform strain as measured by the considered detector. The polarization angle is also uniformly sampled between $[0, 2\pi]$. These computations are carried out by the PyCBC package [58]. The waveform is added to the noise afterward, with a randomly selected start time. The signal is injected after 0.5 s, 1 s, 1.5 s, or 2 s. Before generating the spectrograms, the data are whitened. Spectral whitening is a signal-processing technique that facilitates signal detection in noisy data by equalizing noise amplitude over frequencies. It is carried out by normalizing the time series Fourier transform by its estimated ASD. The last processing step is implemented to reduce strong noise feature effects visible in spectrograms. Indeed, these noise features appear as large spikes in detectors' ASD (cf. Figure 1.3). They produce high-energy horizontal lines in spectrograms, which may or may not be

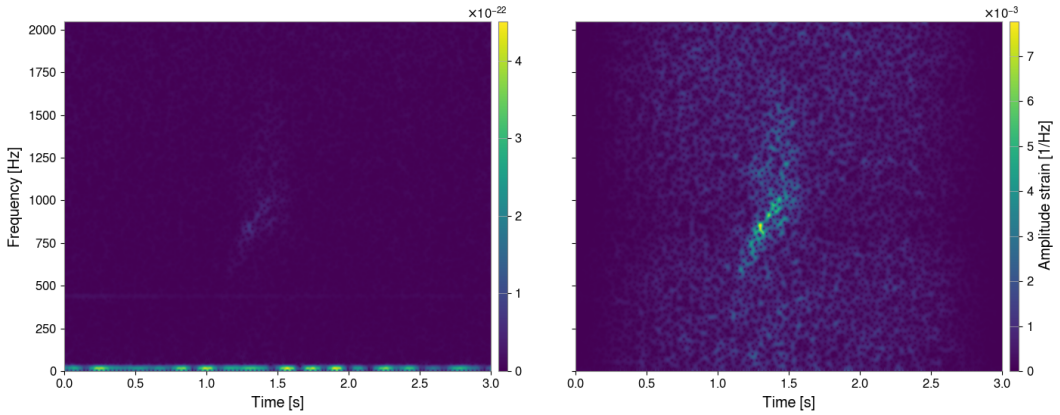

Figure 4.6: Spectrogram before (left panel) and after (right panel) time series spectral whitening and row-wise normalization of the spectrogram. These TF maps display the *s18* model.

continuous. Since their origins are well studied and mostly known (for some details, cf. Section 1.2), we perform a row-wise normalization to artificially tone the lines down. Figure 4.6 illustrates the combined effect of whitening and row-wise normalization on an injection spectrogram. We also enhanced the signal contrast with the background by saturating the pixel magnitude above 99.5% of the highest spectrogram strain magnitude value. An instance of dataset time-frequency representations for each CCSN model and the noise is shown in Figure A.1 and A.3. The dataset spectrograms are finally saved with a resolution of 48×257 pixels (for 3 s $\times$ 2048 Hz) in Red-Green-Blue (RGB) images. RGB images reduce the memory cost compared to Numpy arrays [6]. Note in Figure 4.1 that the UNet's input has three channels, corresponding to the RGB channels.

The overall dataset is created by injecting a given number of waveforms per model at increasing visual SNR to mimic various gravitational wave amplitudes. The visual SNR characterizes how the injection stands out from the noise. We define it as the sum of the difference map pixel values contained in the injection mask. After a visual inspection, we could set a visual SNR lower limit to ensure ASBUS is trained on eye-visible injection. Otherwise, we would indicate to the network the presence of a signal through the target map without ASBUS being able to identify it in the input TF map. In addition, we define a visibility scale specifying the visual SNR at which the waveforms are injected. The scale must realistically reflect the recurrence of GW signal amplitude in practice. Hence, more spectrograms are injected at low visual SNR, and the visibility range covers amplitude from $\sim 10^{-23}$ to $\sim 10^{-20}$.

4.4 Training

Training procedure

Our neural network is trained with a specific aim namely, retrieving pixels of potential CCSN gravitational wave signal. In this framework, *i.e.* ASBUS is designed to output the pixel magnitude of the identified signal. The ground truth values for those pixels are retained in the associated target map. As the anomaly map (noted A), ASBUS output, and the target map (noted T) must be as close as possible, we use the Mean

Square Error (MSE) as a loss to compare the two:

$$MSE = \frac{1}{2}\sum_{i,j}(T_{i,j} - A_{i,j})^2, \tag{4.3}$$

where the indices i, j denote the map pixel (i, j). The target map pixels for noise spectrograms are set to zero.

The chosen optimizer is Adam [39] with a learning rate set at 0.0001. Batch normalization is incorporated as in Boudart implementation and progressive learning is handled via curriculum learning. ASBUS learns first on high visual SNR TF maps, where the signals are easier to detect, and continues on decreasing visual SNR samples. The training and validation sets are balanced between the spectrogram classes (*i.e.* composed of 50% background spectrograms and 50% injection spectrograms). We set the batch size to 32 spectrograms and ran the training for 30 epochs.

ASBUS was first trained on a small dataset consisting of four CCSN waveforms and 10,000 TF maps. We used the latter training to confirm the good operation of the modifications introduced in the algorithm code ensemble. The next training are based on 13 CCSN models described in Section 2.3. Including more models allows for greater diversification of the patterns and characteristics to be learned from the TF maps. If the dataset is too simplistic, the neural network cannot generalize meaningful information out of the training set.

We perform a training on $10,000$ spectrograms per class and the SNR visibility scale used is detailed in Table 4.2. We label the training *ASBUS_10k* hereafter. The dataset is divided into a training set and a validation set with proportions of 90% and 10%, respectively.

We also created a dataset by creating 100 injection spectrograms per CCSN model and injection visual SNR. ASBUS version trained on this dataset is named ASBUS_18k. We changed the visibility scale used in ASBUS_10k for better distribution of the visual SNR. The visibility scale used is referenced in Table 4.2. We obtained a dataset amounting to $\sim$ 36000 samples. In addition, we used the PyTorch function *RandomErasing* as a data augmentation technique, which randomly selects a rectangular area in the input map and erases its pixels. The spectrograms have a 50% chance of being affected by one or 2 rectangular masks. The validation set comprises 20% of the dataset.

Table 4.2
Visibility scale used for injecting CCSN waveforms at a given visual SNR. It is referenced for ASBUS training version on the $10,000$ or $18,000$ injection spectrograms dataset.

Training version	Visibility scale
ASBUS_10k	{450, 1400, 2350, 3300, 4250, 5200, 6150, 7100, 8050, 9225, 10400, 11575, 12750}
ASBUS_18k	{450, 602, 806, 1080, 1446, 1936, 2592, 3471, 4647, 6223, 8332, 11156, 14937, 20000}

We will evaluate ASBUS' version performance on the validation set via the Anomaly Score (AS) [6], which is defined as

$$AS = \sum_{i,j} A_{i,j}, \quad if \;\; A_{i,j} > 0.5\, max(A) \tag{4.4}$$

with $A_{i,j}$ denoting the anomaly map pixel (i, j). This score characterizes the strength of the "anomaly" detected in the input spectrogram. The anomaly refers to the signal recovered in the anomaly map, which could correspond to either a GW injection or a noise-induced signal. Additionally, the score is suitable for distinguishing between a noise anomaly and an injection one. To obtain complementary results, we computed the AS on the target maps, resulting in a target anomaly score, as well as on the pixels of the anomaly map superimposed by the target map mask. The latter score is denoted as the target mask anomaly score. The comparison between the AS and the target anomaly score will inform about ASBUS' ability to recover the whole injected signal and the injection pixel values. The comparison with the target mask anomaly score will reflect ASBUS's ability to localize anomalies relative to the signal in the target map. The scores computed for the injection spectrograms of the validation set are detailed in the following section. We will also describe the computed losses for the training and validation sets, which help us monitor ASBUS's training.

Results

The training and validation losses 18k and 10k versions are shown in Figure 4.7. We can see that the validation losses stabilize over the epochs. Hence, the training could have been stopped earlier, and around 15 epochs were sufficient to train ASBUS. The validation losses do not show the characteristic increase that would indicate network overfitting. However, the loss values drop significantly during the first epochs and decrease slowly afterward. This is understandable because the complexity of the Gaussian is less than the complexity of the true interferometer noise. As a result, ASBUS can easily distinguish the injected signal from the noise. The validation losses end up with a value below 0.5. Additionally, the losses of the 18k model are lower than those of the ASBUS_10k model, indicating a higher signal recovery ability.

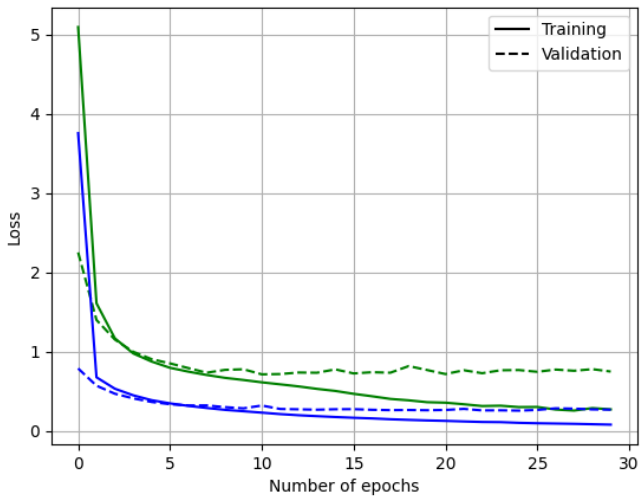

Figure 4.7: Losses resulting of ASBUS training on the dataset of $10,000$ (in green) and of $18,000$ (in blue) injection time-frequency maps. The training (solid line) and validation (dashed line) losses are displayed for both dataset versions.

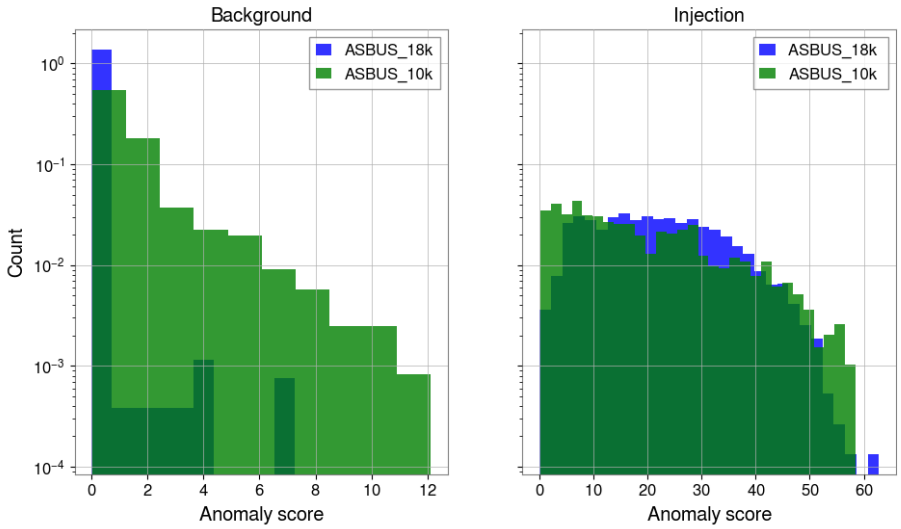

Figure 4.8: Anomaly score distribution over the injection and background spectrograms of the validation set for ASBUS' training version. The histogram bins are normalized.

As a next step, we compute the anomaly scores, as well as the target and target mask anomaly scores, on the validation set. The AS distribution over the dataset classes is shown in Fig. 4.8 first panel. We observe that ASBUS detects anomalies in background spectrograms which is discussed in more detail in Section 4.5. The ASBUS_10k model detects noise-induced anomalies with higher anomaly scores and in greater numbers than the ASBUS_18k model. The maximum AS values are 12.12 and 7.26 for ASBUS_10k and ASBUS_18k, respectively. In theory, the best model would detect no anomalies in noise TF maps. In practice, the fewer anomalies detected with the lowest AS, the better. Moreover, anomaly scores corresponding to injection anomalies have higher values than background anomalies. Hence, given a defined AS threshold, noise- or injection-induces signals could be ranked.

We plotted in Figure 4.9 the target and target mask anomaly scores as a function of their respective anomaly score to compare further both training versions. The scores are calculated on the injection TF maps of the validation set. We notice in the left plot that ASBUS_18k points are gathered along the plot diagonal, indicating ASBUS_18k generates anomaly strength values close to the target map signal strength. This means that the 18k version may recover the whole injection pixels with the correct values. For the same score value, 18k may also output a lower number of pixels with higher pixel values or a higher number of pixels with lower pixel values. Conversely, ASBUS_10k points are more spread in the upper-diagonal region of the plot. This indicates that the 10k version tends to identify signals with a lower strength than the ground truth signal. In this case, the number of pixels, the pixel amplitude values or both of the recovered pixels will cause a reduction in signal strength. The right panel of Fig. 4.9 brings complementary observations. Concerning ASBUS_18k, the majority of the points align along the plot diagonal. It implies that all or almost all the pixels associated with the detected anomaly are localized within the target mask. A few anomalies with AS below ~ 10 are incorrectly localized within the map (*i.e.* presenting a null target mask anomaly score). Overall, the 18k version accurately retrieves the position of the input signal. For the 10k

version, many points are spread below the diagonal for $AS < 30$. It translates to a lower ability to identify the correct signal or to localize it accurately. This is reinforced by seeing that several points align along the x-axis, more than for ASBUS_18k.

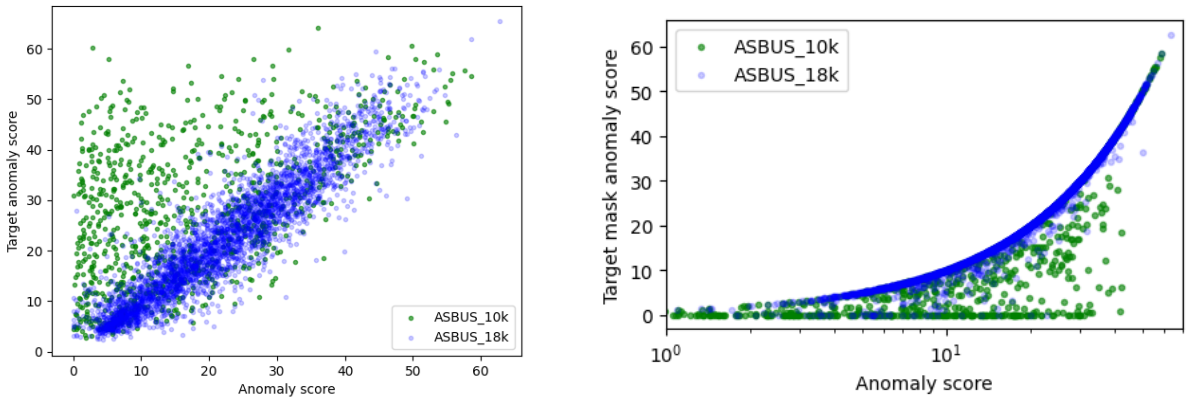

Figure 4.9: Left panel: Plot of target anomaly score, computed over the injection class of the validation set, as a function of the respective anomaly scores. Right panel: Plot of the target mask anomaly score versus the respective anomaly score, with both scores computed on the validation set injection spectrograms.

The observations made based on the losses seem to indicate that the model trained on 18,000 injection TF maps performs better than the 10k model. Combining the results from Figures 4.8 and 4.9 , ASBUS_18k seems to identify the signal present in the input spectrogram in most injection cases while having no detection in a large majority of background TF maps. The 18k version is capable of accurately localizing the identified signal and retrieving its strength, with only a minor number of detections deviating from that accuracy.

We complement the discussion with a visual inspection of ASBUS performance. Figure 4.10 shows an example of injection spectrograms with its associated anomaly map. Other examples are provided in Appendix B. At visual SNR below $\sim$ 450, neither model detects any signals. This is not surprising since the injections are not visible at such values, and the models are trained to recognize signals above a visual SNR value of 450. As the injection strength increases, the 10k model begins to detect the higher strength waveforms (*e.g.*, *s13*, *s25* or *D15-3D* models) at visual SNR around 1400. However, they are poorly recovered. Concerning ASBUS_18k, injections are found in the noise starting from visual SNR values of about $\sim$ 700. In general, the ASBUS_18k anomaly maps are more accurate than those of the 10k model. The visual inspection confirmed the hypothesis that ASBUS_18k is more accurate in reconstructing the injection. Indeed, the 10k model often localizes the pixels in the anomaly maps poorly and has a lower detection capacity. Therefore, only ASBUS_18k will be considered in the following discussion.

Focusing on this version, we noticed that some ASBUS_18k' outputs show split signals. Even if the network is trained to extract pixels as a single cluster, this occurs when only some signal parts stand out from the noise (cf. Fig. B.3). In other cases, when the signal shows significant variability in amplitude strain combined with the low resolution of the TF maps, injections are cut into several pieces. In addition, the detection of models with distinct arcs in TF maps, such as Pan models, is impacted by the method used to compute the target map (cf. Fig. B.1 and Fig. B.2). Indeed, only one signal arc or the beginning of multiple-arc signals is retrieved in anomaly maps. This induces a lower capacity for retrieving multiple GW

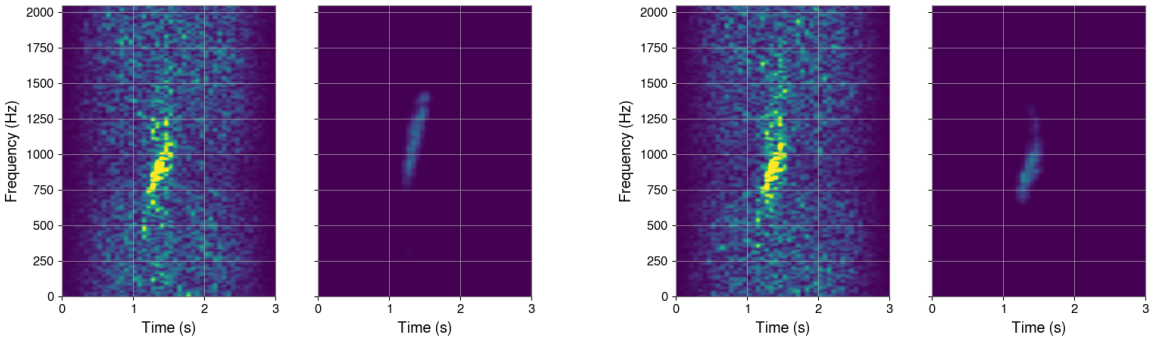

Figure 4.10: Time-frequency map of *s18* model with its anomaly map. The waveform is injected at a SNR value of 100 (*i.e.*, at a visual SNR of ~ 3500). The left panel is the ASBUS_10k output, and the left is the output generated by ASBUS_18k.

features in TF maps. We also observe that at high visual SNR $\sim 20,000$, the injection dominates the noise and the whole signal stands out. In this case, only the main part of the signal is recovered, which is also explained by the idea behind the target map generation.

Note that the *SFHx* evolution in TF maps should display an arcuate evolution. Our signal is folded horizontally from ~ 500 Hz, resulting in a GW feature from 0.2 s to the end of the simulation between 200 and 400 Hz. It could be due to the resampling applied to the simulation time series. While the resulting shape would increase the dataset diversity of GW signatures, its shape resembles noise-induced signals at low visual SNR. As we are working in Gaussian noise, the strongest noise artifacts appear in the TF maps as point-like anomalies. This may have increased the number of detections in the background spectrograms. Although *SFHx* impact on the network performance may not be significant in our case, it would increase the rate of false detections for a search realized in real interferometer noise. The TF shape is similar to several noise transients present in real noise.

For reference, visual SNR values of ~ 400, ~ 2000, ~ 3500 and $\sim 20,000$ correspond to SNR values of ~ 10, ~ 50, ~ 100 and ~ 1000 respectively. Note that these values are associated with the *s18* model and that visual SNR and SNR values are waveform and noise dependent. The Signal-to-Noise Ratio (SNR) is defined by the expression

$$\left(\frac{S}{N}\right)^2 = 4\int_0^\infty \frac{|\tilde{h}(f)|^2}{S_n(f)} df, \tag{4.5}$$

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

In conclusion, ASBUS_18k has a better performance than ASBUS_10k. The increase in the dataset size, the use of data augmentation, and the change in the visibility scale result in a model that retrieves the signal strength and localizes it with accuracy. Henceforth, the ASBUS training version that will be considered is ASBUS_18k and will be referred to as ASBUS.

4.5 Pyxel

The global objective underlying ALBUS is to develop a GW detection pipeline leveraging the processing speed of deep learning algorithms. Once trained, ALBUS represents a component of that pipeline, named Pyxel and developed by Maxime Fays. ALBUS would enable daily data analysis with detection results generated within a few minutes. Pyxel would point out potential GW candidates for other pipelines to analyze and serve as an early warning pipeline for astronomers observing potential event counterparts.

Pyxel workflow begins with the preprocessing of the interferometer time series. The data are whitened and time-frequency representations are generated. Then, their frequency bins are normalized over time and the resulting TF maps are provided as input to ALBUS. While ALBUS acts as a noise removal filter, Pyxel completes a clustering on ALBUS' anomaly map creating triggers. The latter are defined by their respective trigger masks and describe identified signals in ALBUS' input maps. The clustering method corresponds to the following operation sequence: a local thresholding method, an Euclidean distance transform, and a single threshold application. The first threshold uses Yen's method [59]. The method locally adapts the threshold value depending on the noise level. The optimal threshold is selected to divide the image into foreground (the signal of interest) and background. The second operation, namely the Euclidean distance transform attributes to each background pixel the distance to the closest foreground pixel cluster (highlighted by Yen's threshold). Lastly, a given threshold is set on the ensuing image, connecting all pixels with an Euclidean distance below its value. Subsequently, triggers belonging to the same event are not split into several pieces. It maximizes the likelihood of detecting a GW candidate trigger based on detection statistics above the background event statistics. The final clustering map contains the trigger mask created after the Euclidean distance transform threshold. The trigger masks, superimposed on ALBUS' outputs, play the role of generating the final triggers. Pyxel finishes by computing relevant statistics for each trigger. The chosen statistics will provide a global picture of the detection as well as helping to determine the trigger credibility such as gravitational waves candidate.

As ALBUS has been designed to fit into Pyxel workflow, ASBUS is already shaped to be incorporated and has not been modified further. As for Pyxel, some implementation modifications have been made: TF maps computation has been adapted, the Euclidean distance transform has been fine-tuned for signals of hundreds of milliseconds, and the single threshold is set to 0. A non-zero threshold would result in the sum over the background pixels, and given the small spectrogram resolution, the trigger properties and statistics would be inaccurate.

Concerning detection statistics, we use the anomaly score, which we define as follows

$$AS_t = \sum_{i,j \in mask} A_{i,j}, \tag{4.6}$$

for a given trigger t. The indices i, j denote the anomaly map A pixel (i, j) included in the trigger mask. As Boudart demonstrated, this score is suitable for differentiating noise- and signal-induced triggers [6]. The score referenced as the significance by Boudart will not be used since it corresponds to a combination of signal properties designed for thin and long signals.

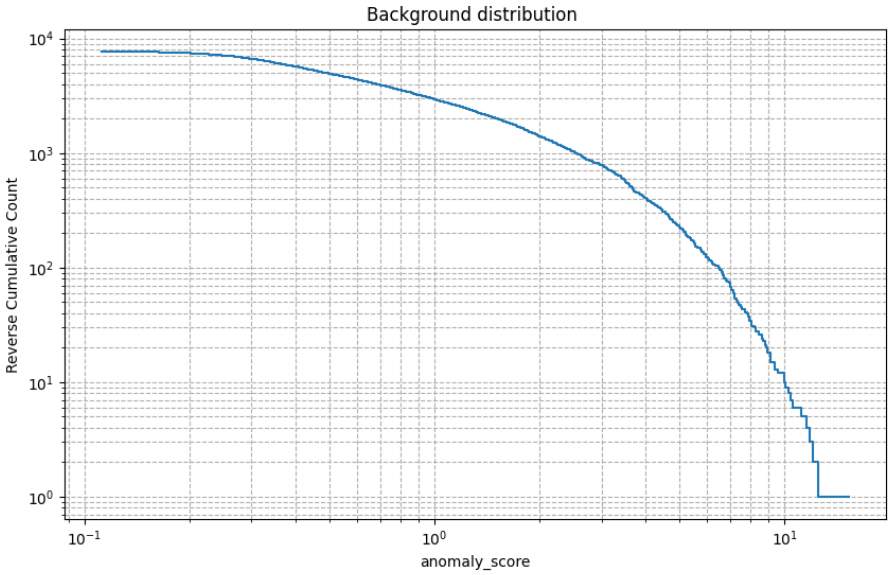

Figure 4.11: Background distribution equivalent to five years of background data.

4.5.1 Background analysis

Any GW search algorithm must estimate the number of false detections that occur over a given period of time. This is referred to as the false alarm rate, which provides an indication of the number of noise-induced triggers that are mistakenly considered to be GW event candidates. These noise-induced triggers are the result of random noise realization or the presence of noise transients. In order to define the false alarm rate and the sensitivity of our algorithm to noise events, it is necessary to process a large amount of data that does not contain a GW signal. This is known as the background. In this project, the background is evaluated by generating a number of background TF maps, with noise generated based on the LIGO design sensitivity, equivalent to five years of data. The resulting reverse cumulative distribution is shown in Figure 4.11. The figure characterizes the number of events with at least an anomaly score value referenced by the curve. It implies that the higher is the trigger anomaly score, the higher is the significance of the trigger. Furthermore, if the anomaly score of a trigger exceeds the AS upper limit (around 15.4), the trigger is more likely to be a GW candidate event. Appendix D displays Pyxel's output for the five loudest background triggers as well as their anomaly score.

Note that in practice the background is estimated over a longer period of time (minimum 10 years up to 100 years) to reduce the statistical uncertainty due to noise fluctuation.

4.5.2 Detection efficiency

We evaluate the sensitivity of the ASBUS search for detecting CCSN waveforms in noise by considering the waveform models listed in Section 2.3. First, the sky positions of the waveforms are randomized to ensure a uniform distribution on a sphere, with the source orientation also uniformly distributed. The waveform is then injected into detector noise (*i.e.*, LIGO design sensitivity) at increasing signal-to-noise ratio values. This

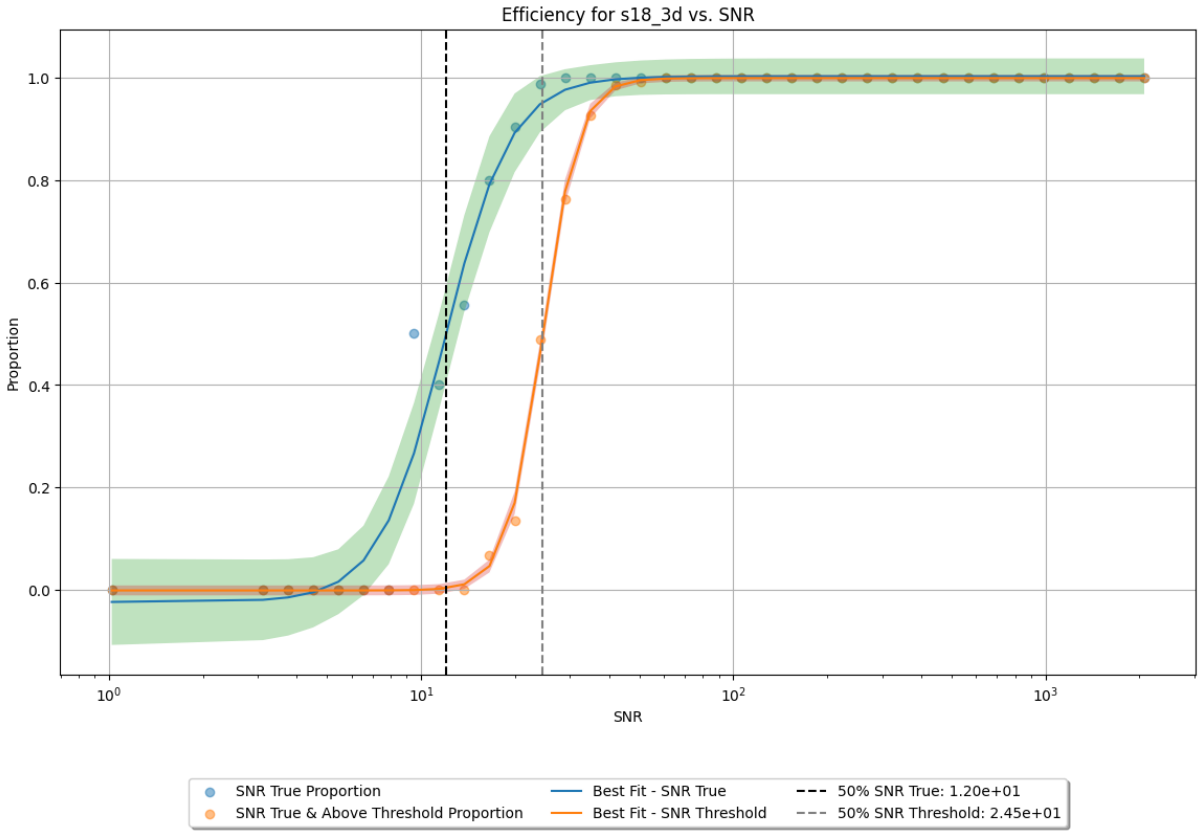

Figure 4.12: Detection efficiency curve of the *s18* model. *50% SNR True* and *50% SNR Threshold* refer to the SNR at 50% detection efficiency for the true proportion and the proportion on which a threshold is set. *SNR True* is the SNR value for the proportion without threshold, and *SNR Threshold* otherwise.

process results in a detection efficiency curve as a function of the SNR. Figures 4.12 shows a result example while Figures 4.13, 4.14 and, 4.15 display the efficiency curves for the other CCSN models. Formally, the search detection efficiency corresponds to the fraction of detected signals with an anomaly score higher than the highest AS trigger value found in the background. We plotted the proportion of detected signals with and without the threshold set on the anomaly score value, represented respectively by orange and blue dots. Note that in the second case, the detected signals are called true proportion.

The considered threshold corresponds to the maximal trigger anomaly score value found in the background. The resulting functions from the fit of a sigmoid function on the data points are plotted in blue for the true proportion and in orange for the proportion with a threshold. The green and orange areas represent the confidence interval at three sigma for each fit. In addition, the 50% detection efficiency is commonly used as a benchmark for typical algorithm sensitivity in GW burst searches. Therefore, the SNR at 50% detection efficiency represents the minimum SNR at which a CCSN can be detected by ASBUS. The latter values are indicated through the dotted lines.

The detection efficiency curves provide information about the search sensitivity. Indeed, the more the 50% detection efficiencies have low values, the more sensitive the algorithm is to the models considered. Besides, the algorithm with an ideal detection capacity would generate equal values for the blue and orange

data points, resulting in no gap between the two proportion curves. This means that the algorithm would only produce high-significance triggers with a statistic value above the background value and the generated triggers would be more likely GW event candidates. The ideal curves correspond to a step function, where the signals are thus recovered above a certain SNR value. In practice, the steeper is the slope of the sigmoid, the better it is. The curves should ideally start from zero and increase to a value of one.

The curve of *s18* model translates a good detection capacity of ASBUS for this waveform. The minimum SNR for ASBUS to detect a *s18* signal equals 24.5. It corresponds to the best minimum SNR value reached for non-rotating models. The rotating model for which ASBUS is most sensitive is the *B12* model which has a 50% SNR value of 29.9. The other values are given in Table 4.3.

Table 4.3

SNR minimal value at which ASBUS is sensitive to the considered waveform model. The CCSN models are described in Section 2.3. The best minimum SNR values for non-rotating and rotating models are shown in bold.

Model identifier	$SNR_{50\%}$ [a]
he3.5	25.8
s18_3d	**24.5**
s13	41.0
s25	28.0
SFHx	- [*]
D15-3D	27.8
mesa20_pert_LR	32.9
Pan_2020_NR	124.0
m39	124.0
s15fr	327.0
Pan_2020_SR	- [*]
Pan_2020_SR	- [*]
B12	**29.9**

[a] $SNR_{50\%}$ corresponds to the SNR value at 50% detection efficiency for detected triggers with an anomaly score above the threshold.

[*] Model with a zero efficiency curve or one that did not reach a proportion of 0.5.

We can see that ASBUS shows good detection efficiency for the rotating models *he3.5*, *s13*, *s25*, *mesa20_pert_LR*, *D15-3D*, and the rotating model *B12*. Although the model *Pan_2020_NR* shows a sigmoid curve with a higher 50% SNR value, the proportion never reaches a value of one. It is also true for *m39* and *Pan_2020_SR*. Indeed, *m39* proportions increase slowly up to 0.7 while *Pan_2020_SR* injections are poorly recovered, with proportions going up to ~ 0.2. Concerning the model *s15fr*, the detection efficiency increases smoothly up to 1 at SNR values above 1000. The lower performance of ASBUS on most rotating models and Pan's non-rotating model may be related to the method of computing the target map. If the waveform TF evolution shows significantly segmented parts, the method will only keep the the part of the signal with the biggest area. In some cases, this will lead to the identification of only a minor part of it.

The development of a method that takes into account multiple GW features in TF maps would increase the sensitivity of ASBUS. On this note, we can also see that some waveform models (*e.g.*, *mesa20_pert_LR*) show a decrease in efficiency at SNR values above $\sim$ 1000. It may be related to the fact that ABSUS is trained to identify signals with lower strength.
Furthermore, the results for *SFHx* and *Pan_2020_FR* are poor, as the detection efficiency curve is almost or constantly zero. Although they have additional or different GW features than the "classical" arc-shaped GW signature, visual inspection of the anomaly maps of both models showed that ASBUS is able to recover at least a part of the injection (cf. Fig. B.1). This may not be sufficient to achieve good detection efficiency. We also encountered computational problems during the search sensitivity evaluation and some of the injections failed. As a result, the apparent poor performance may be due to a lack of data for both models. Ideally, we should have rerun the files that failed in order to assess more precisely the cause of the poor detection on these waveforms.

Overall, ASBUS is sensitive to the majority of non-rotating models with a minimum $SNR_{50\%}$ of 24.5, while it is less efficient for rotating models, except for the *B12* model associated with a $SNR_{50\%}$ of 29.9.

The results obtained from the sensitivity evaluation suggest that ASBUS performs better on waveforms with an arc-shaped signal in TF representations. A solution would have been to train ASBUS on the waveform categories: non-rotating and rotating waveform models. That way, the resulting versions of ASBUS would be specialized in the detection of one waveform type. On another note, conserving *Pan_2020_NR* and *B12* in their respective category for this specialized training would increase the variety of features of each dataset. We would also not include *SFHx* models as the associated TF evolution is too close to noise-induced signals. Looking ahead, it could be interesting to see how the ASBUS version trained on rotating models would recover non-rotating signals, and vice versa.

It is common in CCSN GW searches to express the detection efficiency as a function of source distances. The all-sky search for gravitational waves during O3 run [60] estimated detection ranges of a few kiloparsecs for CCSN gravitational waves emitted by non-rotating models. More powerful CCSN models, such as *m39*, have maximum detection distances of about ten kiloparsecs.

This project opens up avenues for further investigation and potential applications for the future. The next chapter explores some potential prospects.

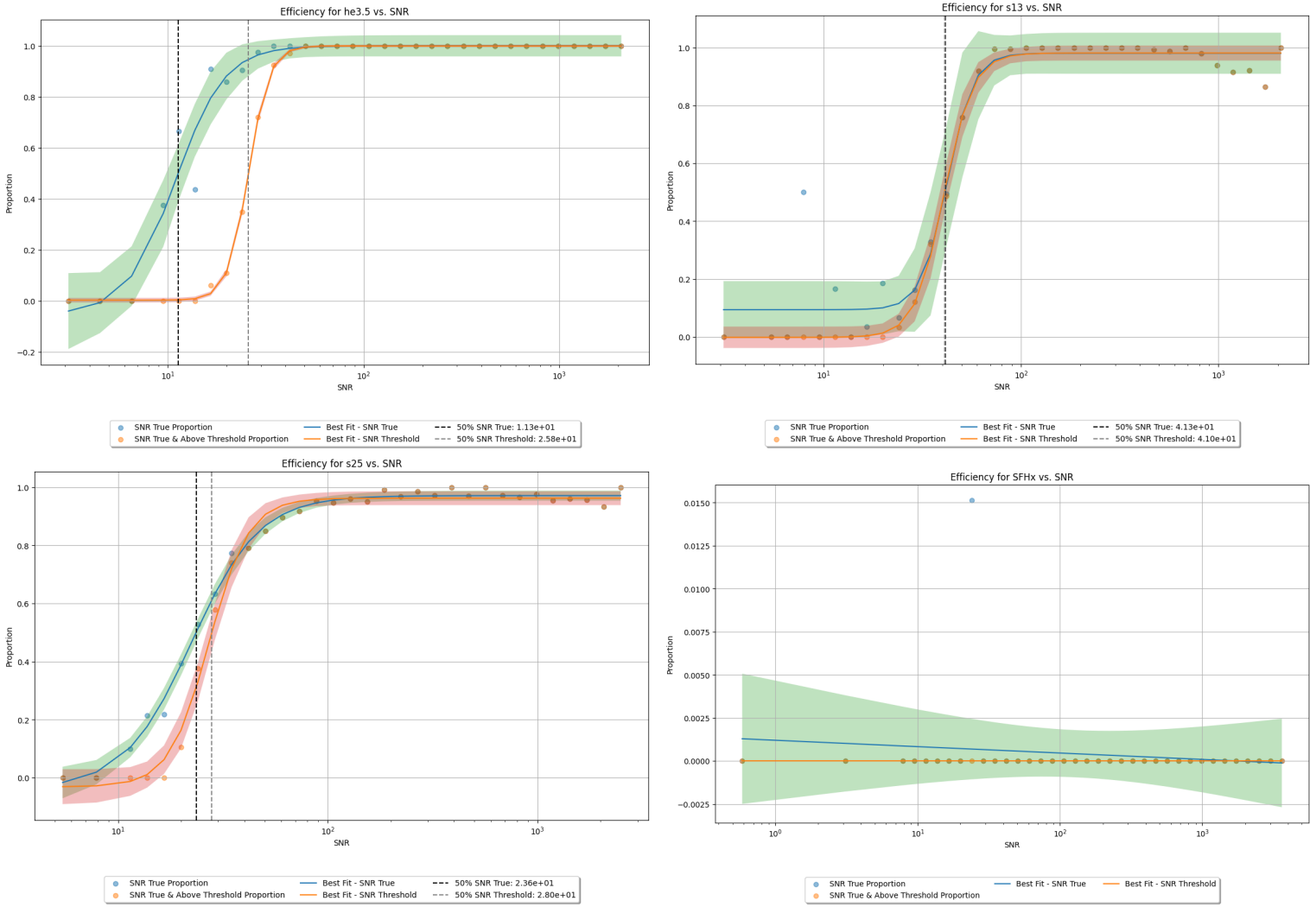

Figure 4.13: Detection efficiency curve of waveform models listed in Section 2.3. *50% SNR True* and *50% SNR Threshold* refer to the SNR at 50% detection efficiency for the true proportion and the proportion on which a threshold is set. *SNR True* is the SNR value for the proportion without threshold, and *SNR Threshold* otherwise.

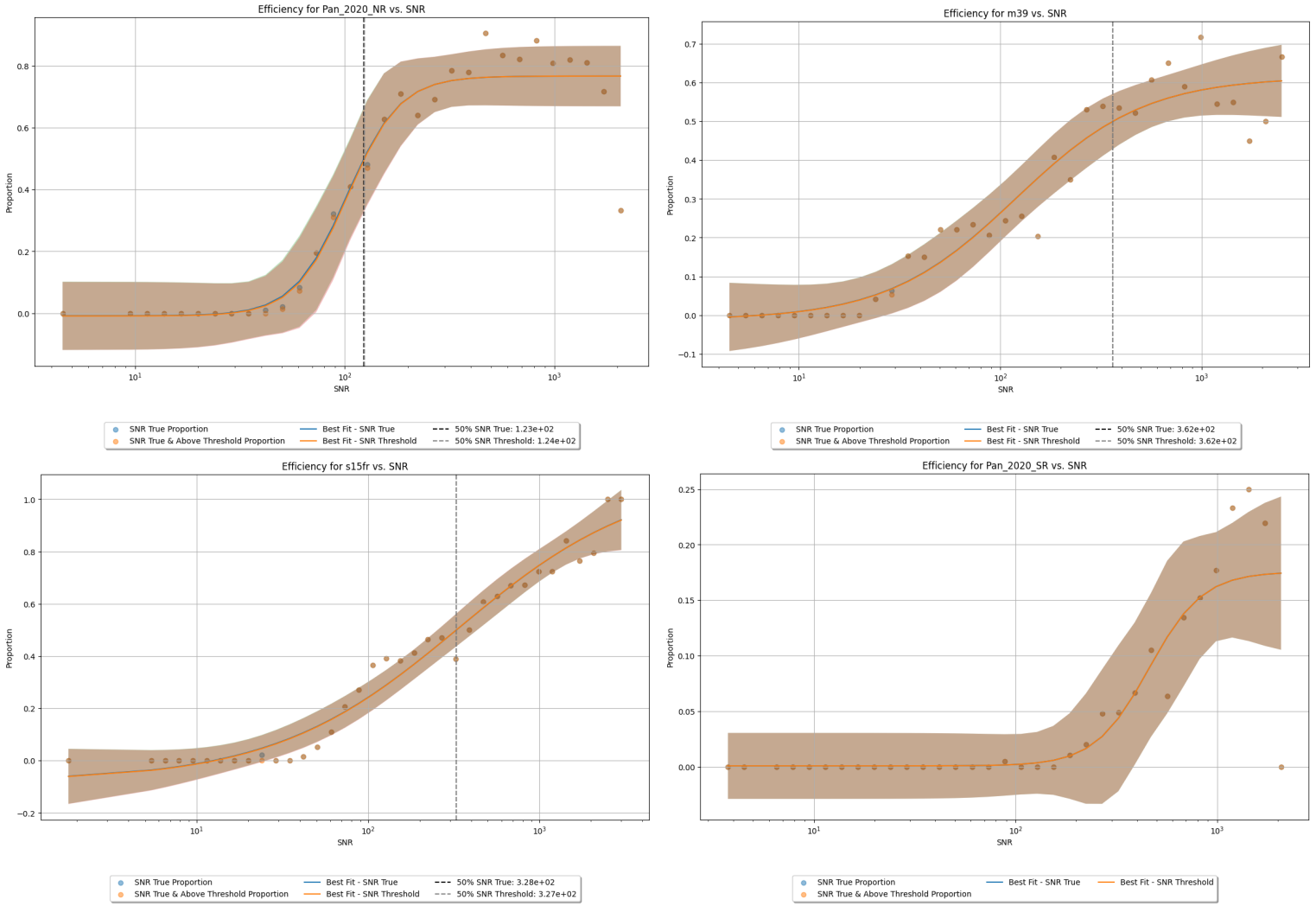

Figure 4.14: Detection efficiency curve of waveform models listed in Section 2.3. *50% SNR True* and *50% SNR Threshold* refer to the SNR at 50% detection efficiency for the true proportion and the proportion on which a threshold is set. *SNR True* is the SNR value for the proportion without threshold, and *SNR Threshold* otherwise.

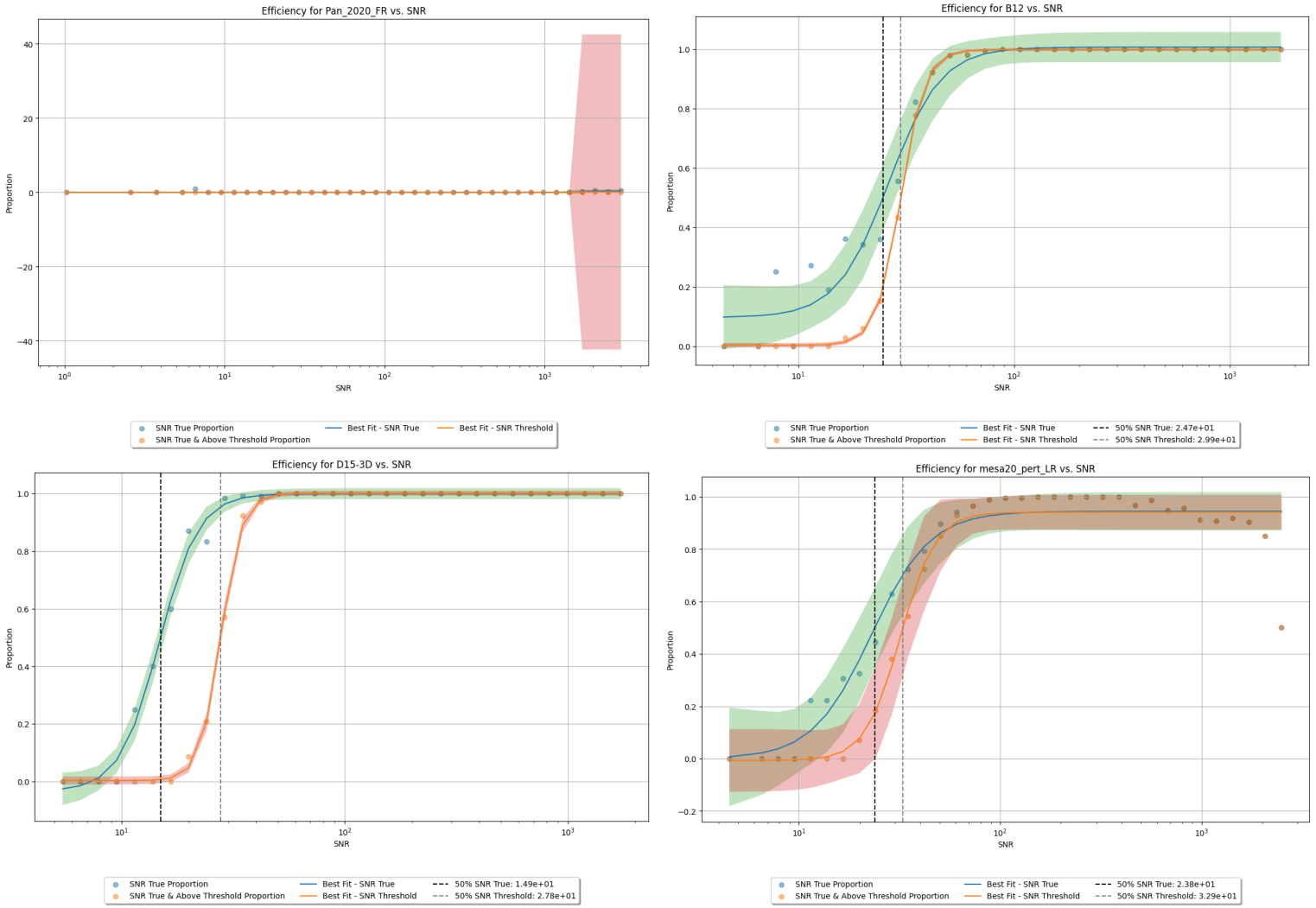

Figure 4.15: Detection efficiency curve of waveform models listed in Section 2.3. *50% SNR True* and *50% SNR Threshold* refer to the SNR at 50% detection efficiency for the true proportion and the proportion on which a threshold is set. *SNR True* is the SNR value for the proportion without threshold, and *SNR Threshold* otherwise.

Chapter 5

Future work and prospects

This research project has enabled us to gain a better understanding of the search for short bursts. Our findings demonstrate that ALBUS is capable, with appropriate adaptations, of detecting short-duration signals from core-collapse supernovae. These findings open valuable perspectives, and several aspects of the project could be further developed and investigated.

Notably, the detections were performed using colored Gaussian noise. This represents the first step in developing a search pipeline. However, interferometer noise must be considered to test ASBUS's performance in a realistic framework. Noise-induced transient signals, called glitches, display various morphology on scales of hundreds of milliseconds (cf. examples Fig. 5.1). A couple of dozen glitch types are described by Gravity Spy [61]. They increase the challenge of GW searches at such time scales and require special care in detecting GW signals in noise. Several tools, such as Gravity Spy or Omicron [62], exist to help identify glitches and thereby reduce the false alarm rate in data. ALBUS's current operating state considers interferometer noise. Instead of predicting a single output map, it generates an anomaly map and a so-called glitch map, encompassing the detected glitches in the input spectrogram. The combination of the ALBUS methodology and the data quality tools mentioned earlier presents a promising perspective for future work.

To further test ASBUS performance, the detection efficiency could have been evaluated on CCSN waveforms not used in the dataset. For instance, the models *y20* [22] and *s18np* [22] could have been utilized as non-rotating models, and the *B10* model [24] as a rotating model. The first and second models simulate the collapse of a 20 $M_\odot$ helium star, which has evolved into a non-rotating Wolf-Rayet star, and a red supergiant star with an 18 $M_\odot$ mass. The third model is equivalent to *B12* model but with a magnetic field strength of 10^{10} G in the core.

In addition, replacing simulated waveforms with phenomenological signals in the training process would increase ASBUS's generalization power. This would result in a more generic dataset, with fewer constraints on signal morphology. Indeed, only a limited number of simulations are currently available, and the parameter space they cover is restricted. The training would be conducted using the phenomenological waveforms, and the performance of ASBUS would be evaluated based on the waveforms from simulations. For instance, the task has been tackled for the neutrino-driven mechanism by Astone *et al.* [49] and later refined by López

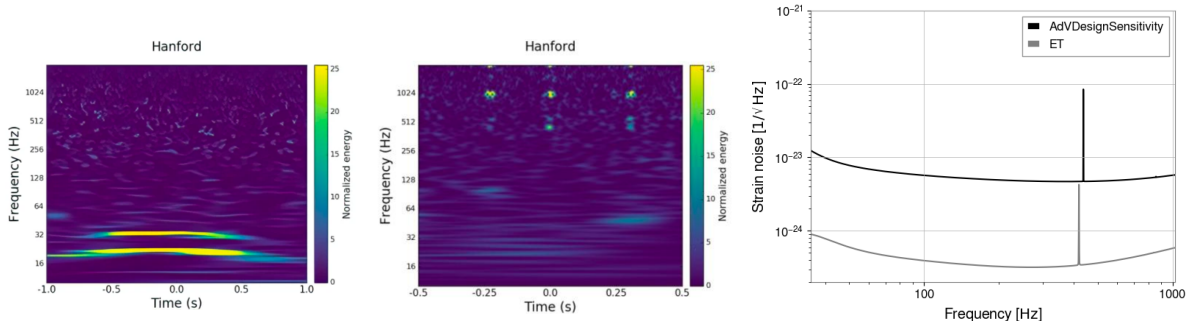

Figure 5.1: Time-frequency representation of scattered light glitch (left panel) and violin modes glitch (middle panel). The right panel compares the Advanced LIGO (AdVDesignSensitivity) and Einstein Telescope (ET) design sensitivity curves.

et al. [53].

Until now, no core-collapse supernova GW has been detected by the LIGO-Virgo-KAGRA collaboration searches [63, 64]. However, the sensibility to those sources is planned to increase with the construction of the underground interferometer *Einstein Telescope* (ET) [65]. While the latter and the LIGO instruments operate in similar frequency bands, ET offers a significantly higher sensitivity. Figure 5.1 compares the design sensitivity of Advanced LIGO and ET. An additional interesting extension of our work would be to explore the distances in the detection efficiency evaluation that could be achieved with ASBUS trained on ET noise. For illustration purposes, the maximum detectable distance for neutrino-driven CCSN such as *s18* model is estimated at 10 kpc and 105 kpc considering respectively Advanced LIGO and ET design sensitivity [16]. Considering a magneto-rotational explosion, Powell *et al.* estimated that the maximum detectable distance for *B12* model is estimated to ~ 200 kpc for LIGO design sensitivity detector and ~ 2 Mpc for ET [24].

Finally, ASBUS could be extended for the detection of short-duration bursts from other sources, such as short gamma-ray bursts or cosmic strings. Moreover, signal detection based on spectrograms is a generic method and has promising applications across several fields. As spectrograms display the time evolution over frequencies of a given signal in some noise, various data types could be represented as such. In biomedical engineering, this method can be used to analyze physiological signals, such as electroencephalography data. Environmental monitoring is another key area, where spectrogram-based detection can help identify specific sounds, such as wildlife calls.

Conclusion

This master thesis set out to develop a deep learning algorithm for the detection of short-duration burst signals from core-collapse supernovae, based on the adaptation of the algorithm ALBUS. The first step involved adapting the time-frequency representation and architecture of ALBUS, followed by the development of a new method for the target map computation. Through these modifications, ALBUS evolved into ASBUS, Anomaly detection for Short-duration BUrst Searches.

ASBUS was trained on waveforms from 3D simulations representing two main explosion mechanisms currently believed to be responsible for core-collapse supernovae. Additionally, two new pixel-based statistics were derived to complement the existing one, enhancing the characterization of signal detection. ASBUS was then incorporated into the pipeline Pyxel for analysis. The results demonstrated that ASBUS is capable of detecting gravitational wave signals from core-collapse supernovae in spectrograms. The algorithm showed higher sensitivity to models associated with neutrino-driven explosions compared to those related to magneto-rotational mechanisms. However, the target map computation was found to be less effective for signals with multiple features in their time-frequency evolution, which are predominant in rotating models. It may be linked to the target map computation. Introducing an enhanced technique that could link split signal parts may increase ASBUS' performance.

Overall, this work has proven that ASBUS is effective in detecting short-duration signals in time-frequency representations. It is hoped that this project will inspire further development of deep learning detection algorithms, leveraging both computation speed and accuracy, not only in the domain of gravitational wave searches but also in other areas of physics.

Bibliography

1. Carroll, S. M. *Spacetime and geometry* (Cambridge University Press, 2019).
2. Mahler, G. *SPAT0012-1 General relativity* University Lecture. 2023.
3. Baltus, G. *A machine learning approach to the search for gravitational waves emitted by light systems* PhD thesis (ULiège-Université de Liège [Sciences], 2022).
4. Habraken, S. *PHYS3032-1 Optique* University Lecture. 2019-2020.
5. LIGO Laboratory Caltech. *LIGO - LIGO's Interferometer* https://www.ligo.caltech.edu/page/ligos-ifo. [Online; accessed 01-August-2024].
6. Boudart, V. *Detection of minute-long Gravitational Wave transients using Deep Learning methods* English. PhD thesis (ULiège - Université de Liège [Sciences], Liège, Belgium, 26 September 2023).
7. Abbott, B. P. *et al.* Observation of gravitational waves from a binary black hole merger. *Physical review letters* **116,** 061102 (2016).
8. LIGO Laboratory Caltech. *LIGO - LIGO Technology* https://www.ligo.caltech.edu/page/ligo-technology. [Online; accessed 01-August-2024].
9. Collaboration, T. L. S. & Aasi, J. Advanced ligo. *Class. Quantum Gravity* **32,** 074001 (2015).
10. Oohara, K.-i., Nakamura, T. & Shibata, M. Chapter 3. A Way to 3D Numerical Relativity. *Progress of Theoretical Physics Supplement* **128,** 183–249. ISSN: 0375-9687. eprint: https://academic.oup.com/ptps/article-pdf/doi/10.1143/PTPS.128.183/5439774/128-183.pdf. https://doi.org/10.1143/PTPS.128.183 (Mar. 1997).
11. Dominique Sluse, M. F. *SPAT0002-1 Statistical methods and data analysis* University Lecture. 2023.
12. Anderson, W. G., Brady, P. R., Creighton, J. D. & Flanagan, E. E. Excess power statistic for detection of burst sources of gravitational radiation. *Physical Review D* **63,** 042003 (2001).
13. Abdikamalov, E., Pagliaroli, G. & Radice, D. in *Handbook of Gravitational Wave Astronomy* 909–945 (Springer, 2022).
14. Szczepańczyk, M. J. *et al.* Detecting and reconstructing gravitational waves from the next galactic core-collapse supernova in the advanced detector era. *Physical Review D* **104,** 102002 (2021).
15. Tauris, T. M., Langer, N. & Podsiadlowski, P. Ultra-stripped supernovae: progenitors and fate. *Monthly Notices of the Royal Astronomical Society* **451,** 2123–2144 (2015).

16. Powell, J. & Müller, B. Gravitational wave emission from 3D explosion models of core-collapse supernovae with low and normal explosion energies. *Monthly Notices of the Royal Astronomical Society* **487,** 1178–1190. ISSN: 0035-8711. eprint: https://academic.oup.com/mnras/article-pdf/487/1/1178/28753300/stz1304.pdf. https://doi.org/10.1093/mnras/stz1304 (May 2019).

17. Radice, D., Morozova, V., Burrows, A., Vartanyan, D. & Nagakura, H. Characterizing the gravitational wave signal from core-collapse supernovae. *The Astrophysical Journal Letters* **876,** L9 (2019).

18. Kuroda, T., Kotake, K. & Takiwaki, T. A NEW GRAVITATIONAL-WAVE SIGNATURE FROM STANDING ACCRETION SHOCK INSTABILITY IN SUPERNOVAE. *The Astrophysical Journal Letters* **829,** L14. https://dx.doi.org/10.3847/2041-8205/829/1/L14 (Sept. 2016).

19. Mezzacappa, A. *et al.* Core collapse supernova gravitational wave emission for progenitors of 9.6, 15, and 25 M. *Physical Review D* **107,** 043008 (2023).

20. O'Connor, E. P. & Couch, S. M. Exploring fundamentally three-dimensional phenomena in high-fidelity simulations of core-collapse supernovae. *The Astrophysical Journal* **865,** 81 (2018).

21. Pan, K.-C., Liebendörfer, M., Couch, S. M. & Thielemann, F.-K. Stellar mass black hole formation and multimessenger signals from three-dimensional rotating core-collapse supernova simulations. *The Astrophysical Journal* **914,** 140 (2021).

22. Powell, J. & Müller, B. Three-dimensional core-collapse supernova simulations of massive and rotating progenitors. *Monthly Notices of the Royal Astronomical Society* **494,** 4665–4675. ISSN: 0035-8711. eprint: https://academic.oup.com/mnras/article-pdf/494/4/4665/33174338/staa1048.pdf. https://doi.org/10.1093/mnras/staa1048 (Apr. 2020).

23. Andresen, H. *et al.* Gravitational waves from 3D core-collapse supernova models: The impact of moderate progenitor rotation. *Monthly Notices of the Royal Astronomical Society* **486,** 2238–2253. ISSN: 0035-8711. eprint: https://academic.oup.com/mnras/article-pdf/486/2/2238/28488247/stz990.pdf. https://doi.org/10.1093/mnras/stz990 (Apr. 2019).

24. Powell, J., Müller, B., Aguilera-Dena, D. R. & Langer, N. Three dimensional magnetorotational core-collapse supernova explosions of a 39 solar mass progenitor star. *Monthly Notices of the Royal Astronomical Society* **522,** 6070–6086. ISSN: 0035-8711. eprint: https://academic.oup.com/mnras/article-pdf/522/4/6070/52241658/stad1292.pdf. https://doi.org/10.1093/mnras/stad1292 (May 2023).

25. Waymo. *Sense, Solve, and Go: The Magic of the Waymo Driver* https://www.youtube.com/watch?v=hA_-MkU0Nfw. [Online; accessed 01-July-2024]. 2022.

26. NVIDIA. *Powering the Future of Clean Energy | I AM AI* https://www.youtube.com/embed/zrcxLZmOyNA. [Online; accessed 01-July-2024]. 2023.

27. Google. *Camels, Code Lab Coats: How AI Is Advancing Science and Medicine* https://www.youtube.com/embed/AbdVsi1VjQY. [Online; accessed 01-July-2024]. 2018.

28. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **5,** 115–133. https://doi.org/10.1007/BF02478259 (1943).

29. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* **65,** 386–408. https://doi.org/10.1037/h0042519 (1958).

30. Louppe, G. *INFO8010 Deep Learning* University Lecture. 2024. https://github.com/glouppe/info8010-deep-learning.

31. Li, H., Xu, Z., Taylor, G., Studer, C. & Goldstein, T. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems* **31** (2018).

32. Bottou, L. & Bousquet, O. in *Optimization for Machine Learning* (The MIT Press, Sept. 2011). ISBN: 9780262298773. eprint: https://direct.mit.edu/book/chapter-pdf/2273493/9780262298773_caq.pdf. https://doi.org/10.7551/mitpress/8996.003.0015.

33. Minsky, M. & Papert, S. A. *Perceptrons: An Introduction to Computational Geometry* ISBN: 9780262343930. https://doi.org/10.7551/mitpress/11301.001.0001 (The MIT Press, Sept. 2017).

34. Glorot, X., Bordes, A. & Bengio, Y. *Deep sparse rectifier neural networks* in *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), 315–323.

35. Maas, A. L., Hannun, A. Y., Ng, A. Y., *et al. Rectifier nonlinearities improve neural network acoustic models* in *Proc. icml* **30** (2013), 3.

36. Clevert, D.-A., Unterthiner, T. & Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).

37. Duchi, J., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* **12** (2011).

38. Geoffrey Hinton, N. S. & Swersky, K. *Neural Networks for Machine Learning. Lecture 6* Coursera. 2018. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

39. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

40. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.

41. Glorot, X. & Bengio, Y. *Understanding the difficulty of training deep feedforward neural networks* in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), 249–256.

42. He, K., Zhang, X., Ren, S. & Sun, J. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification* in *Proceedings of the IEEE international conference on computer vision* (2015), 1026–1034.

43. Ioffe, S. & Szegedy, C. *Batch normalization: Accelerating deep network training by reducing internal covariate shift* in *International conference on machine learning* (2015), 448–456.

44. Geurts, P. & Wehenkel, L. *ELEN062-1 Introduction to Machine Learning* University Lecture. 2023. https://people.montefiore.uliege.be/lwh/AIA/.

45. Fleuret, F. *14x050 Deep learning course* University Lecture. 2021. https://fleuret.org/dlc/.

46. Dumoulin, V. & Visin, F. *A guide to convolution arithmetic for deep learning* 2018. arXiv: 1603.07285.

47. Menéndez-Vázquez, A., Kolstein, M., Martinez, M. & Mir, L. M. Searches for compact binary coalescence events using neural networks in the LIGO/Virgo second observation period. *Physical Review D* **103,** 062004 (2021).

48. Bahaadini, S. *et al.* Machine learning for Gravity Spy: Glitch classification and dataset. *Information Sciences* **444,** 172–186 (2018).

49. Astone, P. *et al.* New method to observe gravitational waves emitted by core collapse supernovae. *Phys. Rev. D* **98,** 122002. https://link.aps.org/doi/10.1103/PhysRevD.98.122002 (12 Dec. 2018).

50. Iess, A., Cuoco, E., Morawski, F. & Powell, J. Core-collapse supernova gravitational-wave search and deep learning classification. *Machine Learning: Science and Technology* **1,** 025014 (2020).

51. Iess, A., Cuoco, E., Morawski, F., Nicolaou, C. & Lahav, O. LSTM and CNN application for core-collapse supernova search in gravitational wave real data. *arXiv preprint arXiv:2301.09387* (2023).

52. Chan, M. L., Heng, I. S. & Messenger, C. Detection and classification of supernova gravitational wave signals: A deep learning approach. *Physical Review D* **102,** 043022 (2020).

53. López, M., Di Palma, I., Drago, M., Cerdá-Durán, P. & Ricci, F. Deep learning for core-collapse supernova detection. *Physical Review D* **103,** 063011 (2021).

54. Sasaoka, S. *et al.* Visualizing convolutional neural network for classifying gravitational waves from core-collapse supernovae. *Physical Review D* **108,** 123033 (2023).

55. Zhang, A., Lipton, Z. C., Li, M. & Smola, A. J. *Dive into Deep Learning* [Online; accessed 01-July-2024]. https://d2l.ai/chapter_computer-vision/transposed-conv.html.

56. Van der Walt, S. *et al.* scikit-image: image processing in Python. *PeerJ* **2,** e453. ISSN: 2167-8359. https://doi.org/10.7717/peerj.453 (June 2014).

57. LIGO Scientific Collaboration, Virgo Collaboration & KAGRA Collaboration. *LVK Algorithm Library - LALSuite* Free software (GPL). 2018.

58. Nitz, A. *et al.* gwastro/pycbc: v2. 3.3 release of PyCBC. *Zenodo* (2024).

59. Yen, J.-C., Chang, F.-J. & Chang, S. A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing* **4,** 370–378 (1995).

60. Abbott, R. *et al.* All-sky search for short gravitational-wave bursts in the third Advanced LIGO and Advanced Virgo run. *Physical Review D* **104,** 122004 (2021).

61. Zevin, M. *et al.* Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science. *Classical and quantum gravity* **34,** 064003 (2017).

62. Robinet, F. *et al.* Omicron: A tool to characterize transient noise in gravitational-wave detectors. *SoftwareX* **12,** 100620. ISSN: 2352-7110. http://dx.doi.org/10.1016/j.softx.2020.100620 (July 2020).

63. Abbott, B. P. *et al.* Optically targeted search for gravitational waves emitted by core-collapse supernovae during the first and second observing runs of advanced LIGO and advanced Virgo. *Physical Review D* **101.** ISSN: 2470-0029. http://dx.doi.org/10.1103/PhysRevD.101.084002 (Apr. 2020).

64. Szczepańczyk, M. J. *et al. An Optically Targeted Search for Gravitational Waves emitted by Core-Collapse Supernovae during the Third Observing Run of Advanced LIGO and Advanced Virgo* 2024. arXiv: 2305.16146 [astro-ph.HE]. https://arxiv.org/abs/2305.16146.

65. Hild, S. *et al.* Sensitivity studies for third-generation gravitational wave observatories. *Classical and Quantum Gravity* **28,** 094013 (May 2011).

Appendix A

ASBUS dataset examples

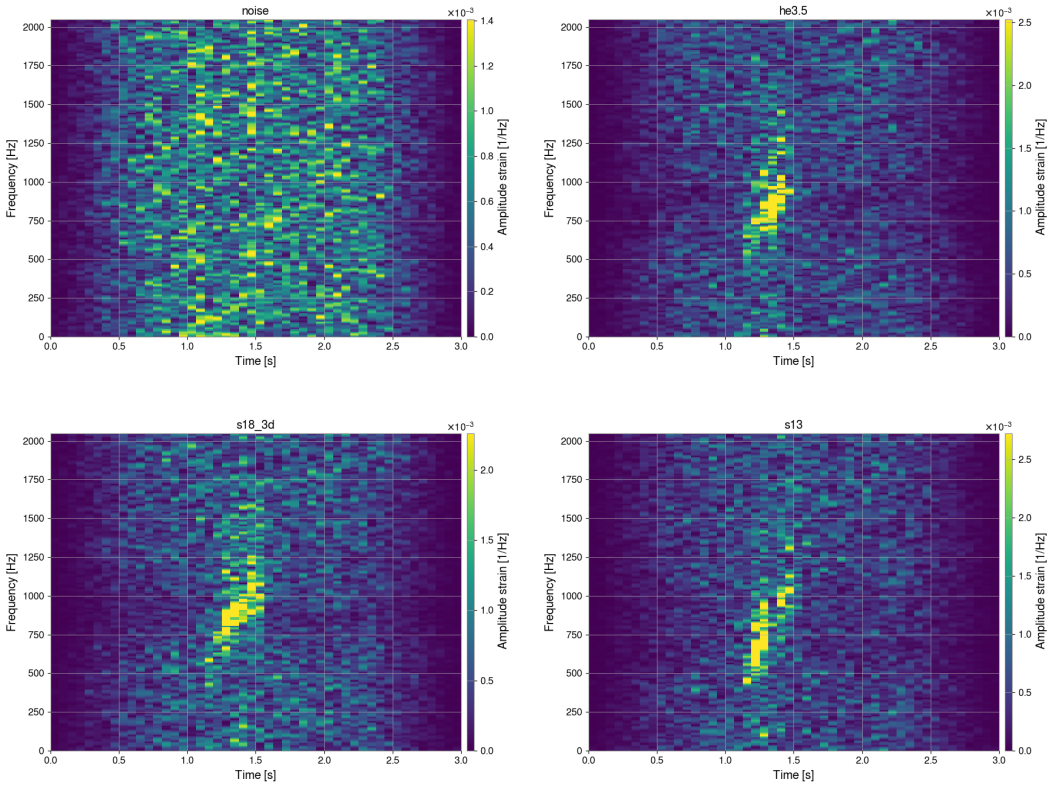

Figure A.1: Dataset spectrogram instances with waveform injected at a SNR value of 100. The CCSN waveform models are listed in Section 2.3. The first panel corresponds to a background spectrogram.

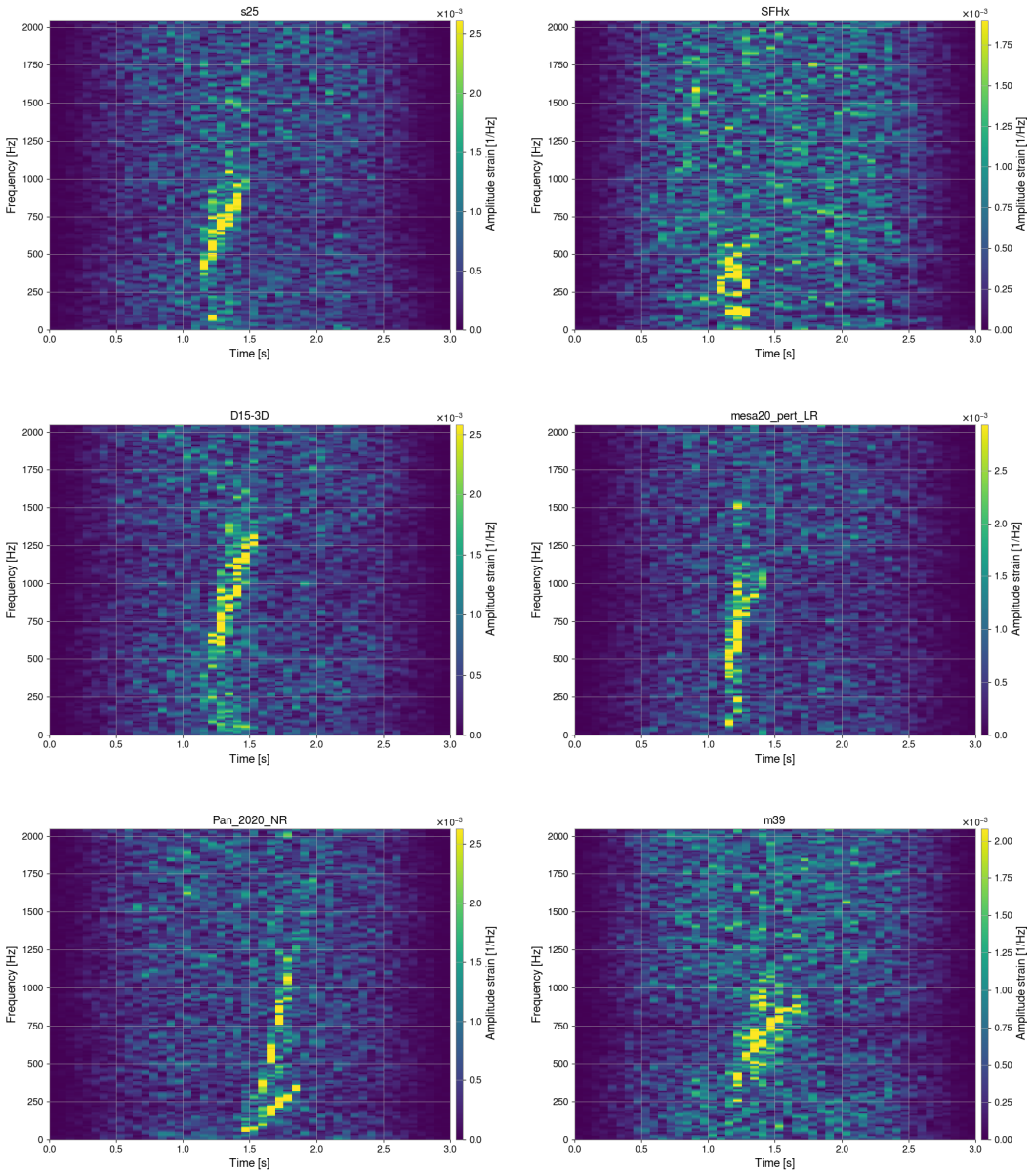

Figure A.2: Dataset spectrogram instances with waveform injected at a SNR value of 100. The CCSN waveform models are listed in Section 2.3.

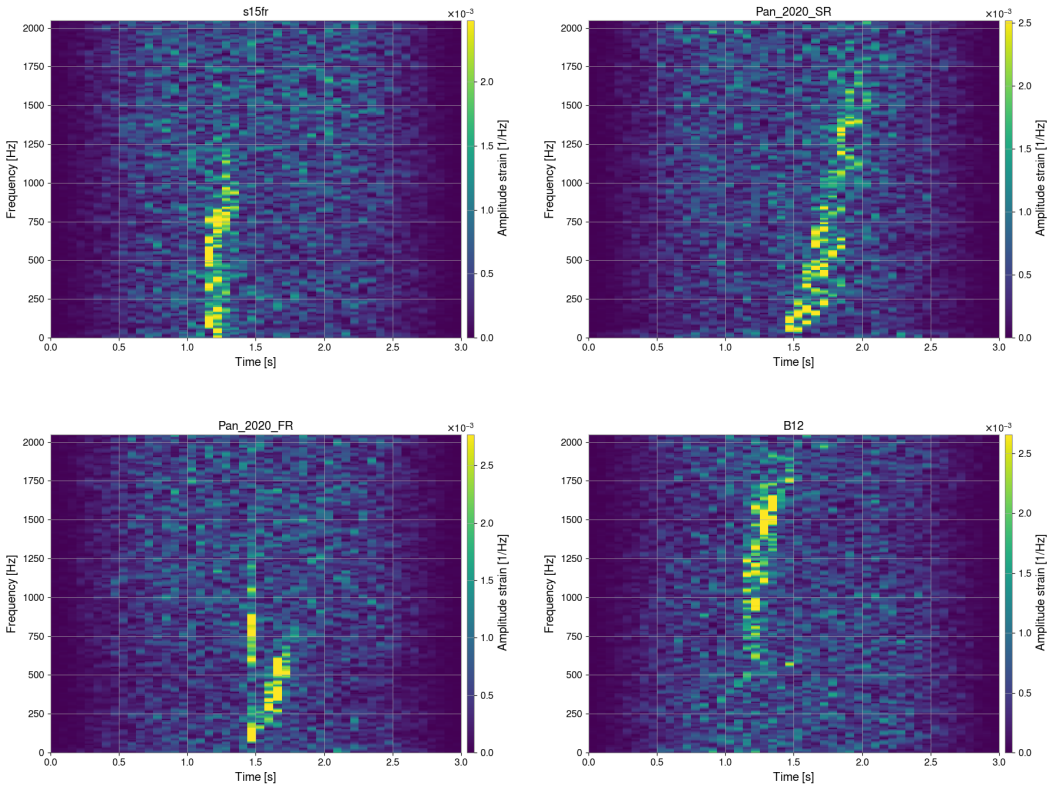

Figure A.3: Dataset spectrogram instances with waveform injected at a SNR value of 100. The CCSN waveform models are listed in Section 2.3.

Appendix B

ASBUS anomaly maps examples

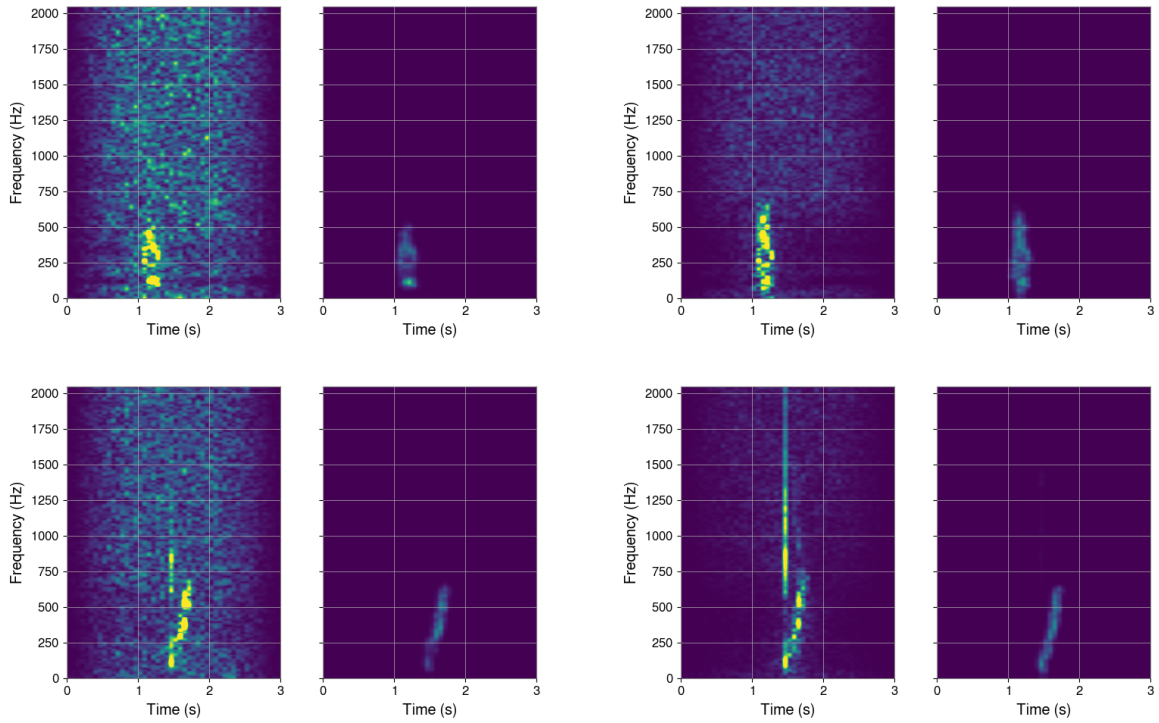

Figure B.1: First row: Injection spectrograms of *SFHx* model with the associated ASBUS_18k's output. Second row: Injection spectrograms of *Pan_2020_FR* with the associated ASBUS_18k's output. The injection SNR values are from left to right: 100, 1000, 100, 1000.

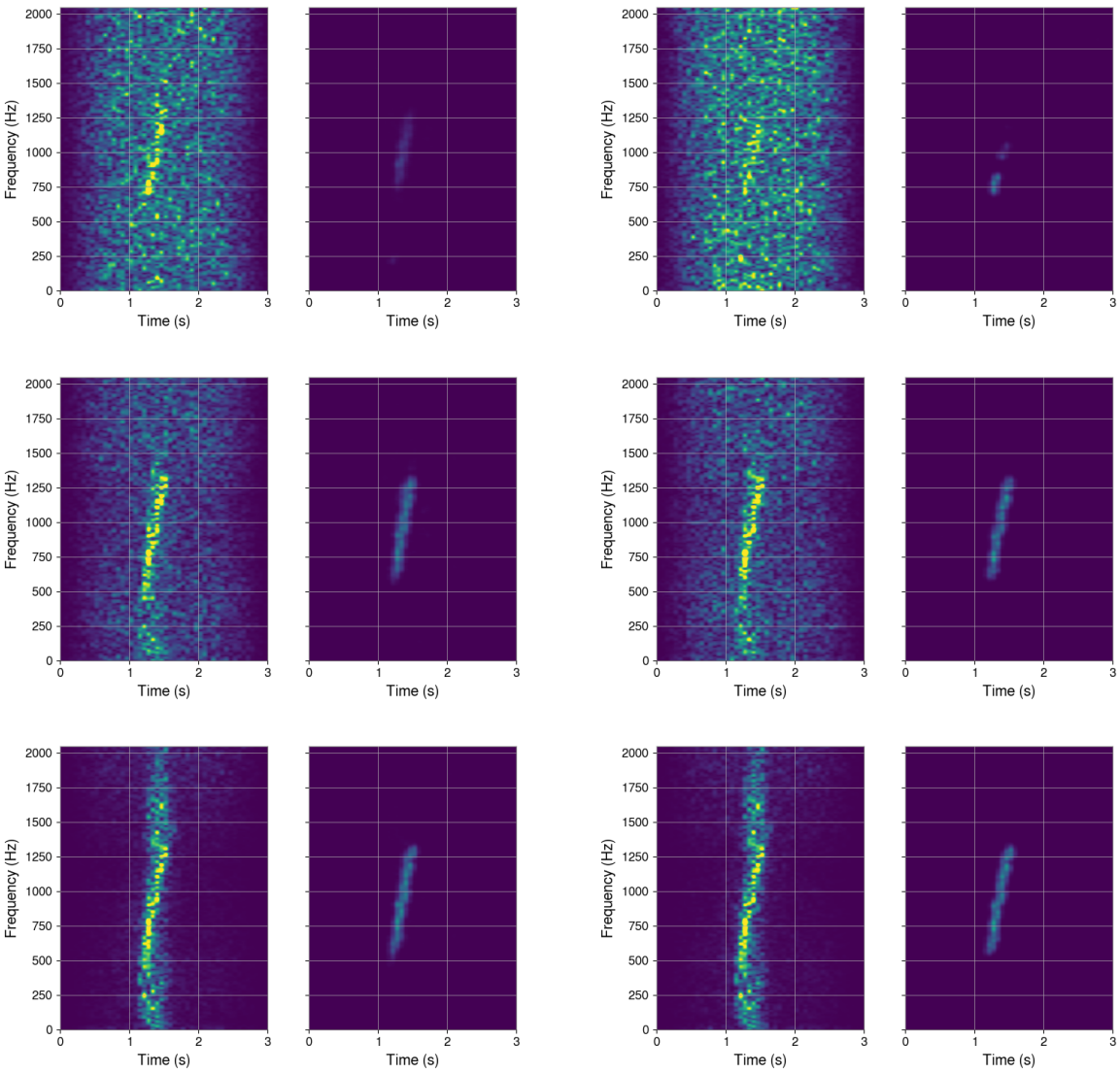

Figure B.2: Left figures: Injection spectrograms of *D15-3D model* with the associated ASBUS_10k's output. Rigth figures: Injection spectrograms of *D15-3D model* with the associated ASBUS_18k's output. The injection SNR values are from left to right: 40, 20, 100, 100, 1000, 1000. The spectrograms with a SNR value of 40 and 20 are the first ten SNR values at which ASBUS detected a signal.

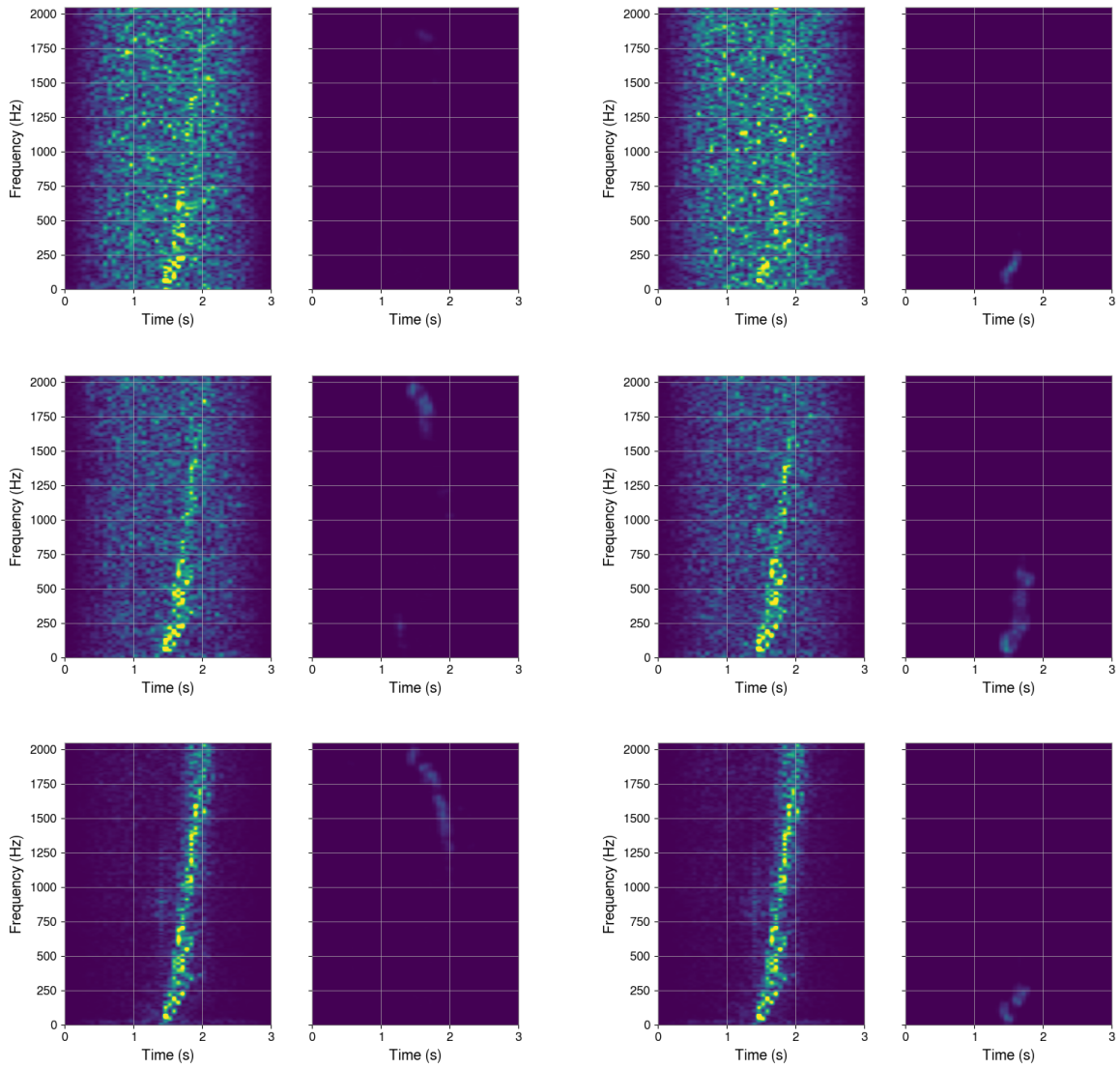

Figure B.3: Left figures: Injection spectrograms of *Pan_2020_SR model* with the associated ASBUS_10k's output. Rigth figures: Injection spectrograms of *Pan_2020_SR model* with the associated ASBUS_18k's output. The injection SNR values are from left to right: 50, 30, 100, 100, 1000, 1000. The spectrograms with a SNR value of 50 and 30 are the first ten SNR values at which ASBUS detected a signal.

Appendix C

Injection mask examples

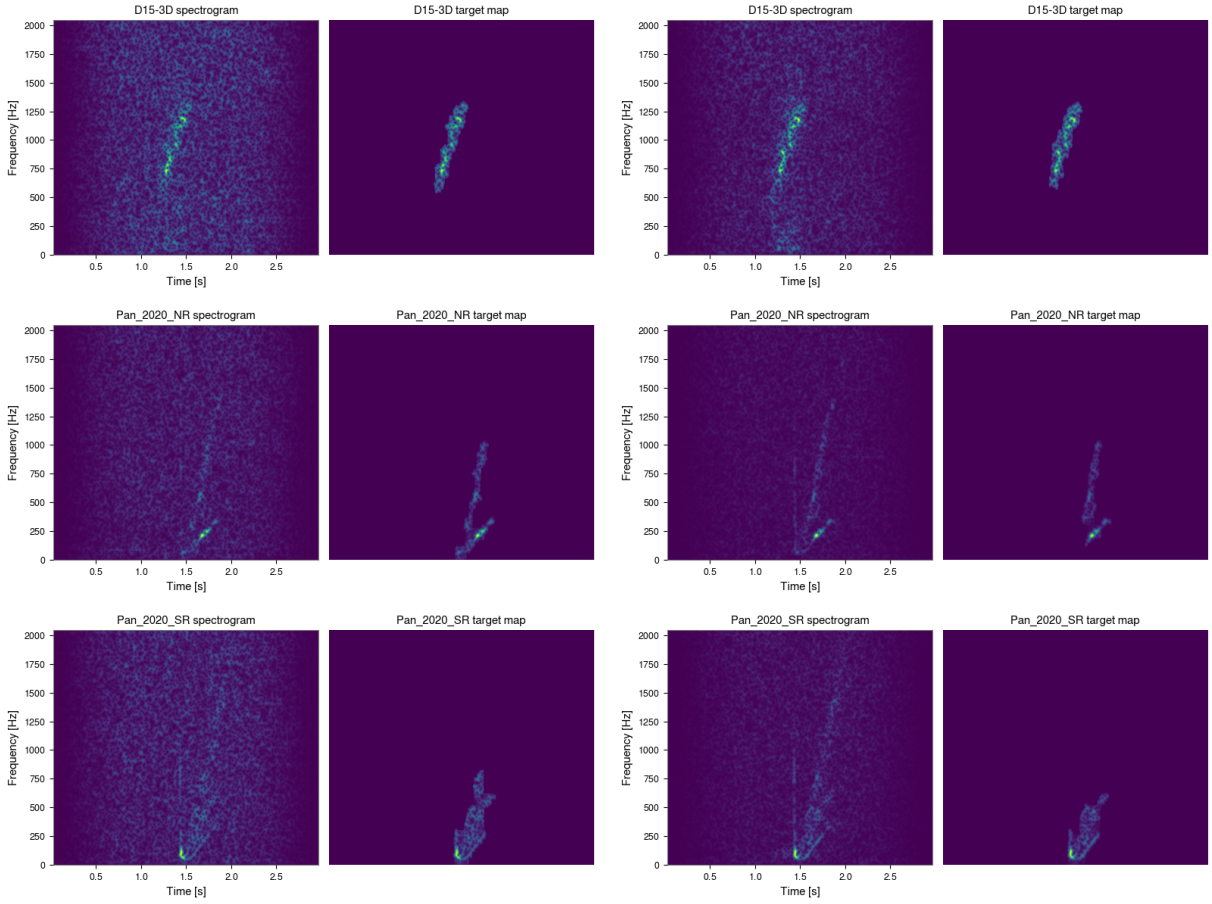

Figure C.1: Injection spectrograms instance with associated target maps. The signals are injected at a SNR value of 50 in the left figures and at a value of 100 in the right figures.

Appendix D

Five loudest background triggers

Table D.1
Trigger properties for the spectrograms containing the five loudest background triggers.

Spectrogram	AS_t	duration [s]	bandwidth [Hz]
1	15.39	0.25	272
2	12.42	0.25	224
3	11.91	0.25	256
4	11.71	0.19	216
5	11.52	0.25	296

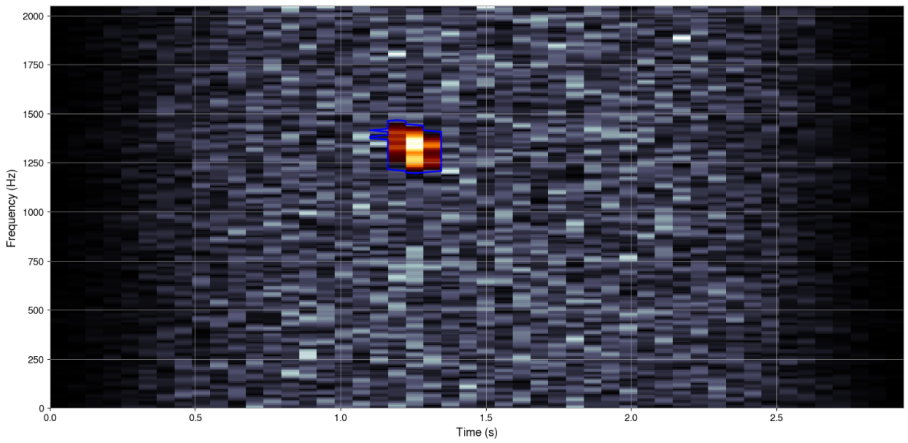

Figure D.1: Noise time-frequency representation (in grayscale). The anomaly map is superimposed in the spectrogram (in orange tones) with the contour of the trigger mask (in blue). The spectrogram is the first loudest background trigger.

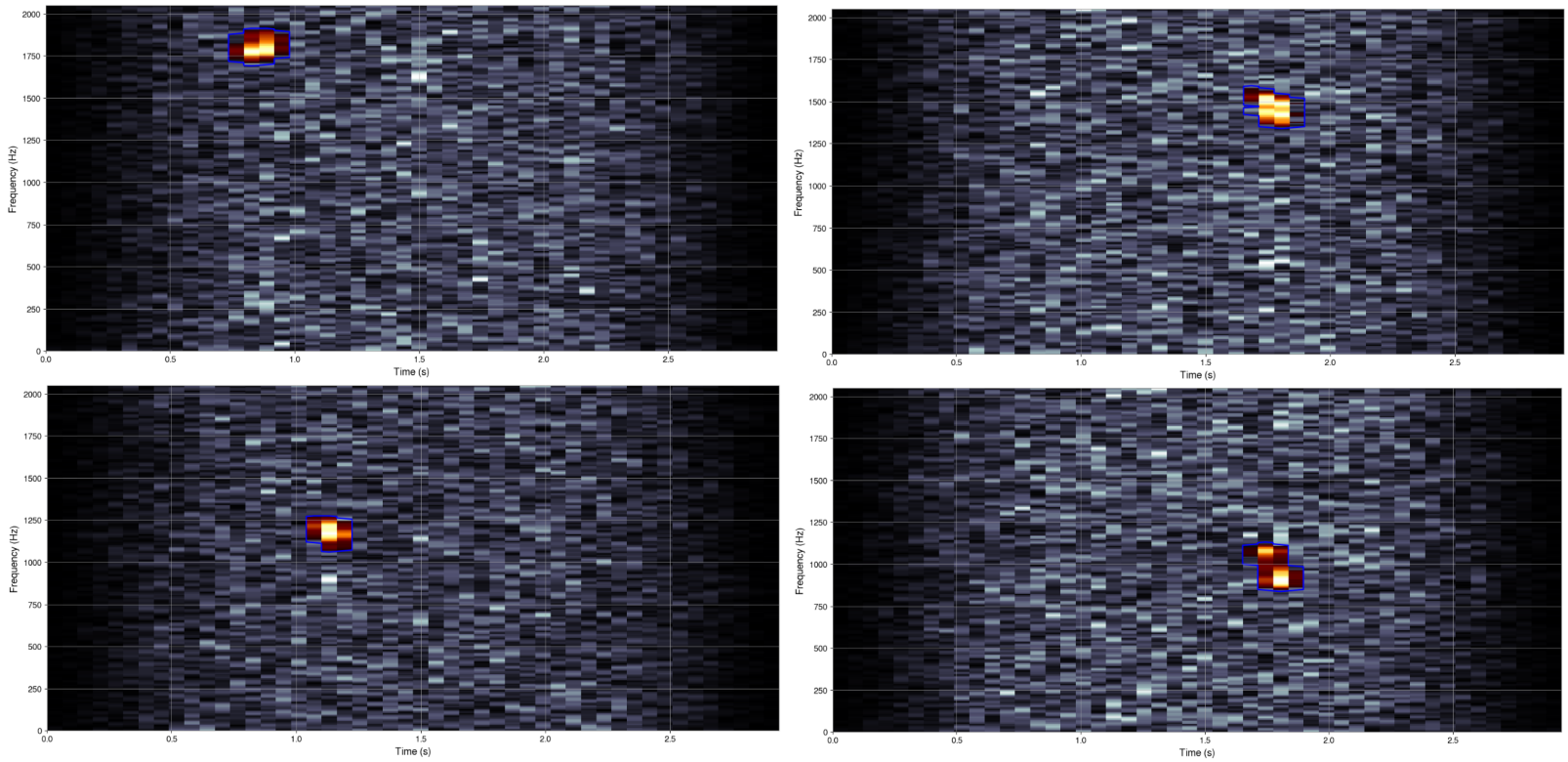

Figure D.2: Noise time-frequency representation (in grayscale). The anomaly map is superimposed in the spectrogram (in orange tones) with the contour of the trigger mask (in blue). The spectrogram are numbered from two to five, from left to right.