UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Cybersecurity and Data Science»

ΠΜΣ «Κυβερνοασφάλεια και Επιστήμη Δεδομένων»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**Rapid Calculation of the Signal-to-Noise Ratio of Gravitational-Wave Sources using Artificial Neural Networks** Γρήγορος υπολογισμός του Signal-to-Noise Ratio των πηγών βαρυτικών κυμάτων χρησιμοποιώντας τεχνητά νευρωνικά δίκτυα
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**Ioannis Tsioulis** Ιωάννης Τσιούλης
Father's name: Πατρώνυμο:	**Athanasios** Αθανάσιος
Student's ID No: Αριθμός Μητρώου:	ΜΠΚΕΔ 21054
Supervisor: Επιβλέπων:	**Dimitrios Apostolou, Professor** Δημήτριος Αποστόλου, Καθηγητής

February 2023/Φεβρουάριος 2023

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Dimitrios Apostolou	**Nikolaos Stergioulas**	**Angelos Pikrakis**
Professor	**Professor**	**Assistant Professor**
Δημήτριος Αποστόλου	Νικόλαος Στεργιούλας	Άγγελος Πικράκης
Καθηγητής	Καθηγητής	Επίκουρος Καθηγητής

Abstract

Using interferometric gravitational wave detectors, we can now observe the dark side of the universe. Already, nearly 90 binary black hole systems have been detected. To measure the source parameters of a detected system, one needs to perform matched filtering, using a large number of templates (of order hundreds of thousand). In such parameter estimation calculations it is also useful to know the optimal signal-to-noise ratio of an individual model waveform. In this thesis, we train an artificial neural network on a random sample of one million theoretical waveforms of binary black hole systems with random spins and achieve an accuracy of 97% in predicting the signal-to-noise ratio. The neural network evaluates the results orders of magnitude faster than the original calculation. We show the results of the optimization of different hyperparameters with a grid search and with selective searches. Finally, we show that the logarithm of the accuracy is linearly related to the logarithm of the number of points in the dataset. This allows us to predict that a dataset size of about 7 million data points will be required to achieve an accuracy of 99% in predicting the signal-to-noise ratio with the neural network that we constructed.

Περίληψη

Χρησιμοποιώντας συμβολομετρικούς ανιχνευτές βαρυτικών κυμάτων, μπορούμε τώρα να παρατηρήσουμε τη σκοτεινή πλευρά του σύμπαντος. Ήδη, έχουν εντοπιστεί σχεδόν 90 διπλά συστήματα μελανών οπών. Για να μετρηθούν οι παράμετροι της ενός συστήματος που ανιχνεύθηκε, πρέπει να γίνει σύγκριση των δεδομένων με πρότυπες κυματομορφές, χρησιμοποιώντας έναν μεγάλο αριθμό προτύπων (τάξης εκατοντάδων χιλιάδων). Σε τέτοιους υπολογισμούς εκτίμησης παραμέτρων είναι επίσης χρήσιμο να γνωρίζουμε τον βέλτιστο λόγο σήματος προς θόρυβο ενός μεμονωμένου μοντέλου κυματομορφής. Σε αυτή τη διπλωματική, εκπαιδεύουμε ένα τεχνητό νευρωνικό δίκτυο σε ένα τυχαίο δείγμα ενός εκατομμυρίου θεωρητικών κυματομορφών διπλών συστημάτων μελανών οπών με τυχαίες περιστροφές και επιτυγχάνουμε ακρίβεια 97% στην πρόβλεψη του λόγου σήματος προς θόρυβο. Το νευρωνικό δίκτυο υπολογίζει τα αποτελέσματα τάξεις μεγέθους ταχύτερα από τον αρχικό υπολογισμό. Δείχνουμε τα αποτελέσματα της βελτιστοποίησης διαφορετικών υπερπαραμέτρων με αναζήτηση πλέγματος και με επιλεκτικές αναζητήσεις. Τέλος, δείχνουμε ότι ο λογάριθμος της ακρίβειας σχετίζεται γραμμικά με τον λογάριθμο του αριθμού των σημείων στο σύνολο δεδομένων. Αυτό μας επιτρέπει να προβλέψουμε ότι θα απαιτηθούν περίπου 7 εκατομμύρια δεδομένα για την επίτευξη ακρίβειας 99% στην πρόβλεψη του λόγου σήματος προς θόρυβο με το νευρωνικό δίκτυο που κατασκευάσαμε.

Acknowledgements

I would like to express my sincere gratitude to my parents for their unwavering support and encouragement throughout my academic journey. Their love, guidance, and sacrifice have been invaluable in helping me achieve my goals. I would also like to extend my heartfelt thanks to my advisors for their guidance, patience, mentorship, and support throughout my thesis research. Special thanks to Prof. Nikolaos Stergioulas, for his insightful feedback and constructive criticism that have been instrumental in shaping my ideas and pushing me to reach my full potential. I am also grateful to Ioannis Liodis for comments on a draft version of the thesis.

Contents

Chapter 1

Introduction

The objective of the project was to build a neural network that predicts the signal to noise ratio (SNR) of gravitational wave sources. This SNR calculation is based on 11 different parameters that describe the physical characteristics of the gravitational wave sources. To train the neural network, 1 million data points were used. The training process resulted in a network that achieved sufficient accuracy for the intended applications. The neural network was implemented using Tensorflow, and the data generation was carried out using the PyCBC gravitational wave libraries, which can be called from a python code.

A black hole is a region of spacetime exhibiting such strong gravitational effects that nothing—not even particles and electromagnetic radiation such as light—can escape from inside it. The theory of general relativity predicts that a sufficiently compact mass can deform spacetime to form a black hole.

The mathematical representation of a black hole is described by the Schwarzschild metric, which is a solution to Einstein's field equations of general relativity. The metric is given by

$$ds^2 = -\left(1 - \frac{2GM}{rc^2}\right) dt^2 + \left(1 - \frac{2GM}{rc^2}\right)^{-1} dr^2 + r^2 d\Omega^2 \tag{1.1}$$

where G is the gravitational constant, M is the mass of the black hole, c is the speed of light, t is time, r is radial distance, and $d\Omega^2$ is the solid angle element. The value $r = 2GM/c^2$ is known as the Schwarzschild radius, and it represents the boundary inside of which nothing can escape the black hole's gravitational pull.

Figure 1.1 is an artist's representation of a binary black hole system radiating graviational waves. A binary black hole system refers to two black holes that orbit each other and are gravitationally bound. These systems are important sources of gravitational waves, which are ripples in the fabric of spacetime. Binary black holes are formed when two massive stars in a binary system collapse into black holes and continue to orbit each other [6].

The data used in this thesis is simulated set of the optimal signal-to-noise ratio (SNR) of gravitational waves received by the LIGO interferometers from coalescing binary black holes. The SNR is a measure of the strength of a gravitational wave signal relative to the background noise in the detectors. Each data point depends on the following 11 parameters:

• Mc (Chirp Mass): This is a combination of the two individual masses that determines the rate of evolution the gravitational wave chirp signal.

• q (mass ratio): This parameter represents the ratio of the masses of the two black holes.

• inc (inclination): The angle between the binary black hole's angular momentum and the line of sight.

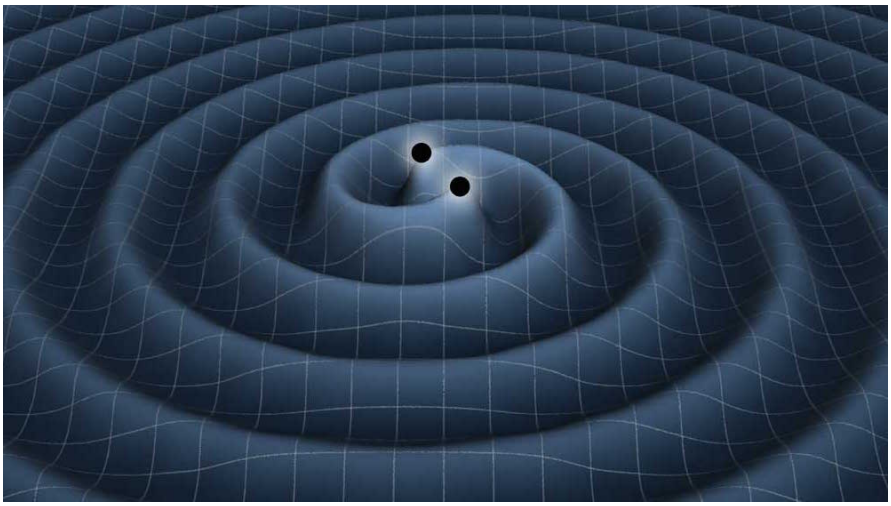

Figure 1.1: Artist's impression of a binary black holes producing gravitational waves that travel outwards and carry energy and angular momentum away from the system [50].

• *ra* (right ascension): The right ascension is the angular distance of an object in the sky from the observer's meridian.

• *dec* (declination): The declination is the latitude-like coordinate that gives the object's position relative to the celestial equator.

• `spin1_mag` (magnitude of spin 1): The magnitude of the first black hole's spin.

• `spin1_polar` (polar angle of spin 1): The polar angle of the first black hole's spin.

• `spin1_azimuthal` (azimuthal angle of spin 1): The azimuthal angle of the first black hole's spin.

• `spin2_mag` (magnitude of spin 2): The magnitude of the second black hole's spin.

• `spin2_polar` (polar angle of spin 2): The polar angle of the second black hole's spin.

• `spin2_azimuthal` (azimuthal angle of spin 2): The azimuthal angle of the second black hole's spin.

The Signal-to-noise ratio (SNR) is a key parameter used in gravitational wave detection to quantify the strength of the signal and determine whether it can be distinguished from the background noise of the detectors. A higher SNR indicates a stronger signal, making it easier to detect and to distinguish from the noise. A lower SNR, on the other hand, implies a weaker signal that may be more difficult to detect. It is essential to have a high SNR for accurate measurement of the properties of the gravitational wave source and to ensure that the detection is significant and not due to random noise fluctuations [4].

The Chirp Mass (M_c) is an important parameter in the analysis of gravitational waves. It is a combination of the two individual masses of a binary black hole system and determines the rate at which the frequency of the gravitational wave signal increases over time. This increase in frequency, known as chirping, is a characteristic of the merging of two black holes. The Chirp Mass is calculated as the product of the total mass of the system and the reduced mass ratio, which is the ratio of the smaller black hole's mass to the total mass.The Chirp Mass is a crucial parameter for determining the characteristics of the binary black hole system and for accurately estimating the source parameters of the gravitational wave signal. It plays a central role in gravitational wave data analysis and can be used to infer the masses and spins of the black holes, as well as the distance to the source [6, 2].

The parameter q or mass ratio represents the ratio of the masses of the two black holes in

a binary black hole system. Mass ratio is defined as the ratio of the mass of the heavier black hole to the mass of the lighter black hole. The mass ratio plays a crucial role in determining the character of the gravitational wave signal emitted by a binary black hole system. It affects the shape and frequency evolution of the gravitational wave signal and can provide important information about the formation and evolution of binary black holes [24].

The inclination (*inc*) is a geometrical quantity that describes the angle between the binary black hole's angular momentum vector and the line of sight from the observer. It is defined as the angle between the normal to the orbital plane of the binary black hole system and the line of sight. The inclination provides valuable information about the orientation of the binary black hole system and helps constrain the binary black hole parameters. The inclination can impact the gravitational wave signal as it is observed and affects the amplitude and phase of the waveform. Accurately determining the inclination is crucial in accurately estimating the parameters of the binary black hole system [5, 3].

In astronomical observations, Right Ascension (*ra*) is a coordinate that specifies the location of an object in the sky, similar to longitude on the Earth's surface. It is defined as the angular distance of the object from the observer's meridian, which is a great circle perpendicular to the observer's horizon and passes through the north and south celestial poles. RA is usually measured in units of time, with one hour of RA equivalent to 15 degrees of angular distance. It is used, along with declination, to specify the position of celestial objects, including stars, galaxies, and black holes. In the context of binary black hole systems, RA is used to determine the location of the source in the sky, which is important for observing and detecting gravitational waves emitted by the system [39].

In astronomical terms, the declination (*dec*) is the angular distance of an object in the sky from the celestial equator, which is similar to latitude on the Earth's surface. The declination is usually measured in degrees, with positive values representing objects located in the northern celestial hemisphere and negative values indicating objects in the southern celestial hemisphere. It is one of the two coordinates, along with right ascension, used to specify the position of an object in the sky. Declination and right ascension together form a celestial coordinate system, which helps astronomers to locate and track celestial objects [49].

The magnitude of the spin of a black hole refers to the strength of its rotational movement. In a binary black hole system, the first black hole's spin magnitude is represented by the parameter `spin1_mag` or `spin2_mag`. This information can be used to better understand the properties of the black hole and the binary system as a whole [66].

The polar angle of spin (`spin1_polar` or `spin2_polar`) is a measure of the orientation of the first black hole's spin vector in three-dimensional space. It represents the angle between the spin vector and a reference direction, typically chosen to be the normal to the plane of the binary orbit. In other words, it describes how "tilted" the spin vector is relative to the binary orbit. This parameter is important for understanding the dynamics of binary black hole systems and can help to constrain the formation and evolution history of these systems.

Azimuthal angle of spin (`spin1_azimuthal` or `spin2_azimuthal`) refers to the angle between the projection of the first black hole's(or the second black hole's) spin vector on the orbital plane and the line connecting the centers of the two black holes in the binary system. This angle provides information about the direction of the spin in the orbital plane. The exact definition and calculation of this parameter in the context of binary black hole systems can be found in studies and articles focused on the modeling and characterization of gravitational wave signals from binary black holes.

Chapter 2

Gravitational Waves

Astronomy has undergone a great revolution, driven by advances in observing capabilities across the electromagnetic spectrum. Within a few years, the first direct detection of gravitational waves has opened up a new window into the universe. The gravitational wave spectrum is completely distinct from, and complementary to, the electromagnetic spectrum. Gravitational waves are emitted by the cumulative mass and momentum of entire systems, so they have long wavelengths and convey direct information about large-scale regions. Given that most of the mass-energy of the universe carries no charge, gravitational waves provide us with our first opportunity to observe directly a major part of the cosmos. Numerous astronomical systems, such as neutron star binaries and cataclysmic variables, are currently better understood owing to the contribution of gravitational radiation theory. Similar to how light and radio waves propagate electromagnetic field oscillations, gravitational waves do the same for the gravitational field. Gravitational waves are produced by accelerating masses as opposed to electrically charged particles, which produce light and radio waves. As the understanding of relativistic events increases, gravitational will likely plays an increasingly important role as a theoretical tool for simulating relativistic astrophysical phenomena [62].

2.1 Gravitational Wave Detectors

The interferometric gravitational-wave detectors with long baselines that were envisioned several decades ago, Laser Interferometer Gravitational-Wave Observatory (LIGO) [1], GEO600 [48], VIRGO [11], have have achieved detections are reaching the required sensitivity levels. Gravitational wave antennas are essentially omnidirectional – in contrast to pointed astronomical antennas and telescopes, their nearly all-sky sensitivity is a key distinction. Antenna patterns typically have a response better than 50% of its average over 75% of the sky. In the next decade, the ability of space-based detectors, such as the planned LISA detector [16], to survey the entire universe for black hole coalescences at milliHertz frequencies will extend gravitational wave astronomy into the cosmological arena. Gravitational wave detectors are more like microphones for sound than conventional telescopes. In a real sense, gravitational wave detectors will listen to the sounds of a restless universe. Gravitational wave detections are allowing us to directly test general relativity in the strongly nonlinear regime. The lower the detector noise, the greater the range of the detectors. A single antenna cannot be used to determine the source's direction. To triangulate the source in the sky by comparing the times of the signal arrival at different detectors in a network, one typically has to observe simultaneously with three or more detectors. The gravitational wave signal arriving at a detector,

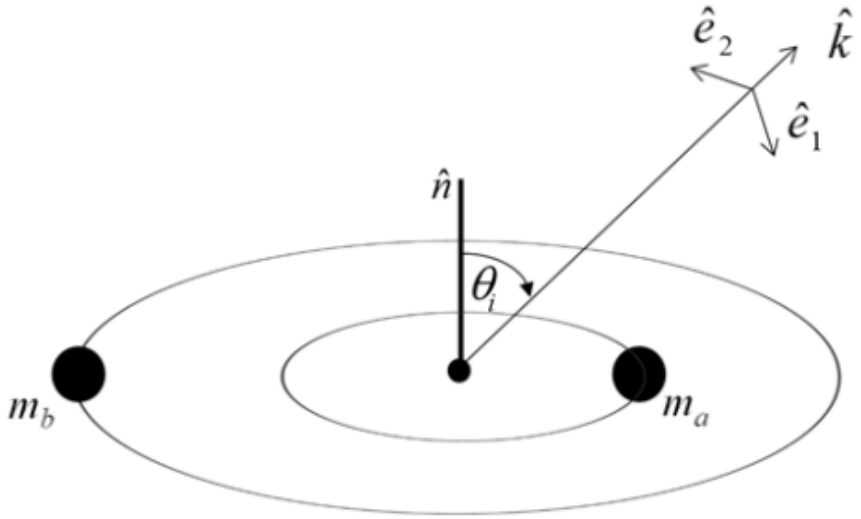

Figure 2.1: Depiction of the orbits of a binary system with masses m_a and m_b, showing how the inclination angle and unit vectors perpendicular to and along the line of sight. Figure from [42].

depends on a number of parameters, such as a set of vectors that describe the detector's geometry, the orbit's inclination angle, and the polarization angle.

In Figure 2.1, we see an illustration of the orbits of a binary system with masses m_a and m_b, illustrating how the angle of inclination θ_i and unit vectors perpendicular to and along the line of sight are defined. The unit vector $\hat{n}$ is parallel to the system's orbital angular momentum and perpendicular to the plane of the orbit. The gravitational wave unit vector $\hat{k}$ directs toward the observation site from the center of mass of the binary systems. $\hat{e}_2$ lies in the $(\hat{n}, \hat{k})$ plane and $\hat{e}_1$ is perpendicular to that plane [62].

Figure 2.2 displays the various factors that affect the measured signal of a binary black holes merger [63]. The diagram consists of three distinct coordinate systems, represented by three differently colored planes. The white plane belongs to the source and is parallel to the orbital plane. The yellow frame represents the preferred reference frame for gravitational waves (GW), where the x_3-axis aligns with the propagation direction of the GW. The deformed ellipsoid on the plane indicates the motion of a ring of test masses in this frame. The green plane shows the detector frame, where the x- and y-axes align with the detector arms and z points away from the center of the earth (depicted as a gray sphere). The angles depicted in the diagram represent the associated transformations. The distance between the detector and the source is represented by the dashed gray line labeled r. The figure does not include the time of coalescence t_0, the coalescence phase Φ_0, or the angular momentum $\vec{L}$. The diagram is not drawn to scale and serves only to illustrate the significance of the various parameters involved in the BBH merger measurement.

2.2 Gravitational Wave Polarizations

Due to the equivalence principle, it is impossible to detect gravitational waves using a single isolated particle because they fall freely in all gravitational fields. A second-rank, symmetric trace-free tensor serves as the gravitational radiation representation in general relativity. This tensor contains 10 independent components in both a general coordinate system and any

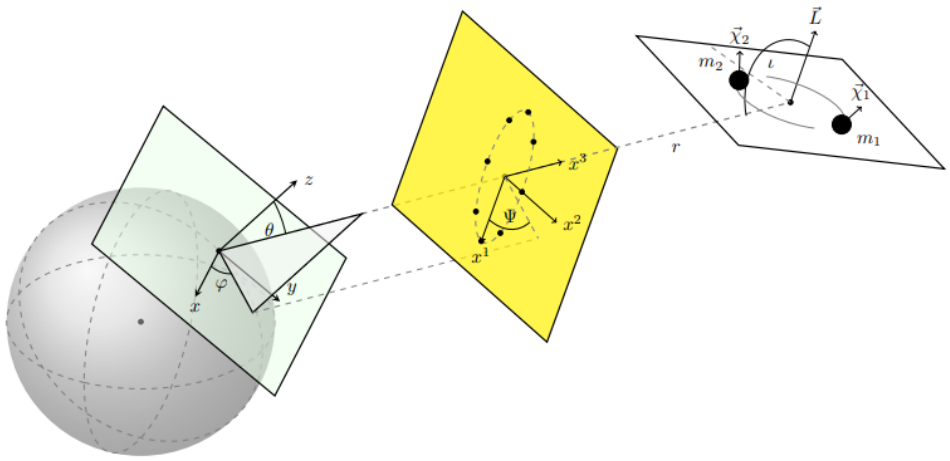

Figure 2.2: A diagram illustrating the parameters that affect the measured signal of a binary black holes merger. The three different coordinate systems used to calculate the GW effect are related to the three differently colored planes. Figure from [63].

gauge (coordinate choice). But according to Einstein's theory, gravitational radiation only has two distinct states of polarization: the plus polarization (h_+) and the cross-polarization ($h_\times$). The result of the waves is that a circular ring is tidally deformed into an elliptical ring, Fig. 2.3. The fundamental idea guiding the design of gravitational wave antennas is the tidal deformation brought on by passing gravitational waves.

In the transverse-traceless (TT) gauge, the gravitational wave tensor can be expressed as:

$$\mathrm{h} = h_+ \mathrm{e}_+ + h_\times \mathrm{e}_\times, \tag{2.1}$$

where e_+ and $\mathrm{e}_\times$ are tensors describing the two polarizations and h_+ and $h_\times$ are the corresponding amplitudes. If a rotation by an angle ψ is applied, the new amplitudes become

$$h'_+ = \cos 2\psi h_+ + \sin 2\psi h_\times, \tag{2.2}$$

$$h'_\times = -\sin 2\psi h_+ + \cos 2\psi h_\times. \tag{2.3}$$

This demonstrates the quadrupolar character of the polarizations. Moreover, a rotation by $\pi/4$ changes one polarization into the other [62].

2.3 Gravitational Waves Frequency

Electromagnetic waves produced by astronomical sources have frequencies that start at 10^7 Hz and continue on upward for close to twenty orders of magnitude. Gravitational waves from anticipated astrophysical sources will likely have detectable frequencies at most 10^4 Hz and and ought to continue lower from there by approximately twenty orders of magnitude [67].

In the case of signals from isolated neutron stars (which have not yet been detected), one can arrive at accurate predictions of the frequencies, even if a pulsar's spin may modulate it

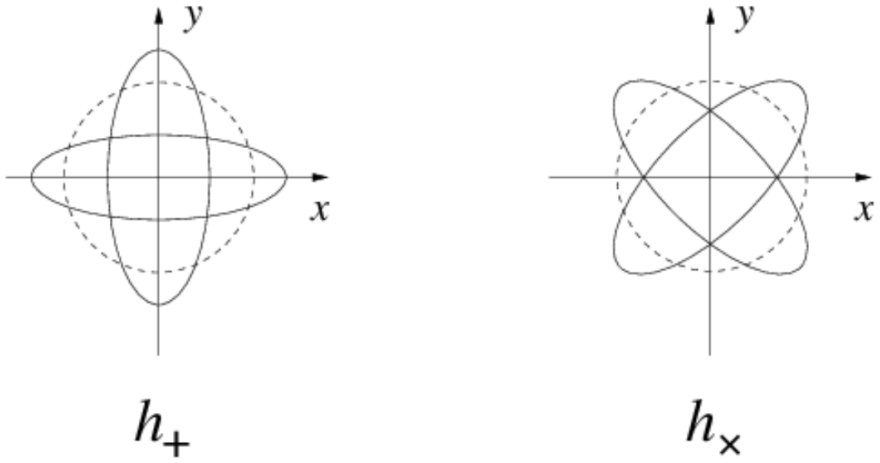

Figure 2.3: The effect of plus-polarized and cross-polarized gravitational waves travelling along the z direction, on a circular ring of matter in the (x, y)-plane [52].

strongly. In the case of oscillating neutron stars, the dominant frequency will correspond to the self-gravitating body's inherent frequency, which is approximately

$$\omega_0 = \sqrt{\pi G \bar{\rho}}, \tag{2.4}$$

or

$$f_0 = \omega_0 / 2\pi = \sqrt{G\bar{\rho}/4\pi}, \tag{2.5}$$

where $\bar{\rho}$ is the mean density of mass-energy in the source. The radius R and mass M of the source define the mean density and, thus, the frequency. The dominant frequency that is characteristic of the radiation that is emitted by a compact object with mass M and radius R is thus

$$f_0 = \frac{1}{4\pi}\left(\frac{3M}{R^3}\right)^{1/2}. \tag{2.6}$$

2.4 Gravitational Waves from Gravitational Collapse and spinning neutron stars

The gravitational core collapse of a highly evolved star or the collapse of an accreting white dwarf produces neutron stars and black holes. Depending on the geometry of the collapse, gravitational waves may transport away some of the binding energy and angular momentum if the collapse is nonspherical (brought on by a large spin). The GW emission mechanism after bounce is similar in the two cases, although slower rotation will have an impact on the likelihood that rotational instabilities will arise.

The method for calculating the rate of a normal supernova may also be used to calculate the rate of a massive star collapse: multiplying the star formation rate by the percentage of stars larger than 20 $M_\odot$. The bottom limit for black hole formation (theory and observations), the initial mass function, and the star formation rate are the uncertainties in such a computation. Fryer and Kalogera [32] estimated that 10–40% of massive stars above 10 $M_\odot$, which are likely to create luminous core-collapse supernovae and contribute to the reported supernova rate, form black holes. Their winds and initial mass function determine this answer. As winds remove less mass from stars, more stars can form black holes at higher redshifts [33].

Some gravitational wave sources behave like the centrifuge, but on a larger scale. A neutron star with radius R, mass M, and a deformation of its otherwise axially symmetric shape, spins with frequency f. Although it will be a mass distribution resulting in an asymmetrical quadrupole tensor, we idealize this as a "bump" of mass m on its surface. The bump's moment of inertia is mR^2, and it is usually parameterized by the fractional imbalance it generates in the tensor. In the near future, it is expected that even signals that are less than the amplitude dictated by the Crab spindown rate will be visible, and these signals may originate from a wider range of neutron stars, especially low-mass X-ray binary systems (LMXBs). They ought to be spinning up because the neutron stars inside them are gaining mass and angular momentum [62].

2.5 Sensitivity of Gravitational Wave Detectors

Interferometric detectors and resonant mass detectors are the two main categories of gravitational wave detectors. In Interferometric detectors, gravitational waves interact with two kilometer-scale arms, their distance being monitored by laser beams. In resonant mass detectors, a heavy body receives energy from the gravitational wave, and the resulting oscillations are then detected. Current ground-based interferometers in use are the two LIGO detectors [1] in the US, the Virgo detector [11] in Italy, the GEO600 detector [48] in Germany and the Kagra detector [14] in Japan. A million-kilometer-scale laser interferometer will also be launched into space with the ESA-NASA LISA mission [16], to observe milliHertz gravitational waves. Furthermore, radio astronomers have also been keeping an eye on faraway pulsars' radio beams for signs of gravity waves for a long time; improved radio technology is making this a potent and promising way to search for stochastic backgrounds and individual sources. Additionally, gravitational waves from the early cosmos may have left their mark on the polarization of the cosmic microwave background at extremely low frequencies.

Joseph Weber [69] constructed two circular aluminum bar detectors and tried to find linked disturbances that might have been induced by a passing impulsive gravitational wave. These detectors were the first type of detector created in the laboratory to detect gravitational waves. Many more bar detectors of equivalent or higher sensitivity were built as a result of his reported detections, but they never corroborated his assertions [62].

The era of gravitational wave detectors was finally initiated in 2015 with the first source GW150914 and in the first three observing runs (O1-O3) of the LIGO-Virgo-Kagra Collaboration a number of $O(90)$ confident gravitational wave (GW) detections were found in the data, which included mostly binary black holes (BBH), but also a few binary neutron stars (BNS) and neutron star black hole systems (NSBH) [7, 8, 9].

2.6 Principles of Resonant Mass Detector Operation

A basic "bar" detector is made of an aluminum cylinder with a length of $\ell \sim 3$ m, a rather small resonance frequency between $f \sim 500$ Hz and 1.5kHz and a mass $M \sim 1000$ kg. A brief blast of gravitational waves with strain amplitute $h \sim 10^{-21}$ will cause an amplitude of vibration in the bar

$$\delta\ell_{gw} \sim h\ell \sim 10^{-21} \text{ m}. \tag{2.7}$$

Three main noise sources must be overcome to measure this [62].

First is thermal noise – the original Weber bar functioned at ambient temperature, but later the ultra-cryogenic Nautilus [17] and Auriga [46] detectors operated at a temperature of T=100 mK. At this temperature, the vibration's root mean square (RMS) amplitude is

$$\left\langle \delta \ell^2 \right\rangle_{th}^{1/2} = \left(\frac{kT}{4\pi^2 M f^2} \right)^{1/2}. \tag{2.8}$$

This exceeds the gravitational wave amplitude predicted by astrophysical sources by a significant margin.

Second is the sensor noise, the mechanical energy of the bar is transformed into electrical energy via a transducer, and an amplifier amplifies the electrical signal to record it. Since both thermal impulses and gravitational wave forces are mechanical forces, the ratio of their produced vibrations would be the same at all frequencies for a given applied impulsive force. If vibration sensing could be done precisely, the detector would be broadband. However, sensing is not perfect, since small amplitudes are difficult to measure due to noise introduced by amplifiers.

Lastly the third main source of noise is quantum noise. A bar's zero-point vibrations at 1 kHz in frequency are

$$\left\langle \delta \ell^2 \right\rangle_{th:1\ \mathrm{ms}}^{1/2} = \left(\frac{kT}{4\pi^2 M f^2 Q} \right)^{1/2} \sim 6 \times 10^{-21}\ \mathrm{m}. \tag{2.9}$$

Because of this, as detectors go over their thermal limits, they must also overcome the quantum barrier to detect a signal at 10^{-21}. There is a chance that one could surpass the quantum limit. The limit above is only imposed by the uncertainty principle when a measurement aims to ascertain the phonon number, which is the equivalent of the excitation energy of the bar. However, the phonon number is only relevant insofar as it can be used to estimate the gravitational wave's initial amplitude. By reducing their uncertainty at the expense of larger errors in its conjugate observable, it is feasible to construct alternative observables that likewise respond to gravitational waves and can be measured with greater accuracy [20]. Squeezing is currently a well-established technique in quantum optics and it is relevant for interferometric detectors, but it is not yet apparent if it will be practical for bar detectors. As was evident in the 1970s, bar detectors have a tough time reaching the sensitivity of 10^{-21}. For a similar objective, a spherical prototype known as MiniGRAIL[38] was developed [62].

2.7 Principles of Interferometric Detectors

The difference between the lengths of two perpendicular (or nearly perpendicular) arms is measured by interferometers using laser light. An interferometer is a natural tool to measure gravitational waves because the arm lengths typically react differently to a given gravitational wave. However, other detectors also make use of electromagnetic radiation, such as solar system spacecraft and even pulsar timing. The stability of the clock continues to be a limiting factor for the sensitivity of such a one-path device as a gravitational wave detector. Interferometers are now the most sensitive beam detectors because they effectively transform one arm into a "clock" or at the very least a time reference against which fluctuations in the other arm are measured. Naturally, the interferometer won't detect a gravitational wave if both arms are impacted in the same way. But only in very unique geometries does this occur. A straightforward interferometer calculates the difference between the round-trip travel

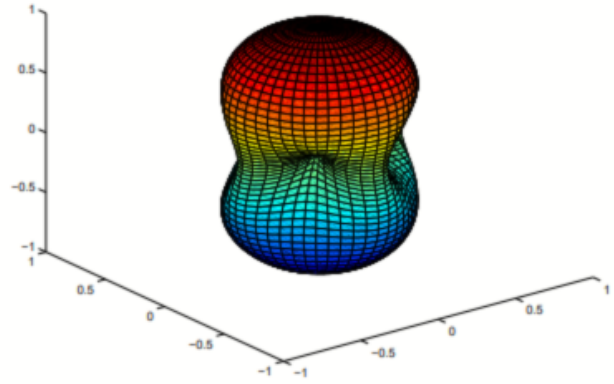

Figure 2.4: An interferometric detector's antenna pattern (left panel), with the arms aligned with the two axes in the x-y plane. The distance to a point on the antenna pattern facing that direction determines the response for waves arriving from that direction [52].

time variations in the two arms for the majority of wave arrival directions and polarizations. Although the measured signal for the triangular space array LISA is a more complicated, the idea that the time reference for one arm is a mixture of the others is still upheld [62].

2.8 Ground-Based Interferometer

The LIGO detectors [56], two of which have arm lengths of 4 km, are the largest detectors currently in operation. One detector cannot simultaneously measure the two separate polarizations of a gravitational wave; instead, it responds to a linear combination of the two, which depends on the detector's geometry and the direction of the source. The responses of three widely spaced detectors, along with two independent discrepancies in arrival times between them, are, in theory, sufficient to completely reconstruct the source location and gravitational wave polarization if the wave only lasts for a brief period. The motion of the detector will cause a long-lived wave to shift where it is in the antenna pattern and modulate its frequency; these effects are, in theory, enough to pinpoint the source's location and the wave's polarization.

In Figure 2.4 the antenna pattern of an interferometric detector is shown, with the arms aligned with the two axes in the x-y plane. The distance to a point on the antenna pattern facing that direction determines the response F for waves arriving from that direction. Over 40% of the sky, the response outperforms the RMS value, suggesting that gravitational wave detectors are largely omnidirectional. Contrarily, just a small portion of the sky is covered by the majority of traditional telescopes (radio, infrared, optical, etc.) [62].

2.9 Problems with Ground-Based Interferometers

The majority of interferometers achieve this duration of light retention in the arms by creating Fabry-Perot cavities, or optical cavities, with low-transmissivity mirrors. A measurement must contend with the following noise sources:

Ground vibration. Mechanical vibrations from the outside must be kept out. These are an issue for bar detectors as well, but they are more problematic for interferometers because interferometers bounce light back and forth between the mirrors, which causes more

vibrational noise with each reflection. These systems, however, can be extremely advanced; for instance, the GEO600 [26] detector contains a three-stage pendulum in addition to additional vibration isolation parts [54]. The most ambitious multi-stage isolation mechanism has been constructed for the Virgo detector [34] [62].

Thermal noise. Mirror and pendulum vibrations can hide gravitational waves. Light bouncing between mirrors increases this, like vibrational noise. Interferometers measure frequencies, unlike bars, distance from the resonance frequency, where thermal vibration is greatest. Current detectors measure above 15 Hz because pendulum suspensions have thermal noise at a few Hertz. Mirror internal vibrations at several kHz limit the observation band. Ensuring that both forms of oscillations have very high Q confines most of the vibration energy to a tiny bandwidth around the resonant frequency, resulting in very small vibration amplitudes at measurement frequencies. Most current interferometers work at room temperature. Thermal disturbances go beyond vibration. Interferometers have transmissive beam splitters and mirrors. Transmission absorbs some light power, raising the mirror's temperature and index of refraction. Time-dependent thermal fluctuations (thermo-refractive noise) can generate random lensing fluctuations at measurement frequencies, destroying the system's optical characteristics. These factors limit detector laser power. Heating effects in multiple-layer mirror coatings can cause other issues.

Shot noise. Light recycling reduces this problem. A power-recycling mirror allows light wasted by a laser to be reflected into the arms of the laser, allowing power to build up in the arms. The first interferometers worked with laser powers of 5 – 10 W, but later upgrades used ten or more times this input power.

Quantum effects. Quantum noises like shot noise have conjugate noises. As laser power is increased to reduce shot noise, position sensing accuracy improves, but the Heisenberg uncertainty principle (momentum imparted to the mirror by the measurement) causes a disturbance that can hide a gravitational wave. Light squeezing is an important technique currently applied to reduced this noise.

Gravity gradient noise. The local gravitational field changes on the measurement timescale. Gravitational wave detectors also detect local tidal forces. Seismic waves cause gravitational field changes, and air pressure changes air density. The spectrum declines rapidly with frequency.

2.10 Optimal signal-to-noise ratio

Data analysis for gravitational wave observation is very different from astronomical data analysis for several reasons:

• Gravitational wave antennas are omni-directional and respond better than 50% of the root mean square over 75% of the sky. Thus, data analysis systems must do all-sky source searches.

• Broadband interferometers encompass three to four frequency orders. This helps us track sources with quickly changing frequencies, but it requires searching a large range of frequencies.

• According to Einstein, gravitational radiation has two polarizations. Measuring polarization is important because there are other theories of gravity with more than two polarization states and even dipolar and scalar waves [71]. It also has astrophysical implications, such as resolving the mass-inclination degeneracy of binary systems observed electromagnetically. A network of detectors can measure polarization, but analysis techniques for various antenna

data must be devised. This should improve coincidence analysis and event recognition efficiency.

• Most astrophysical gravitational waves are detected coherently by tracing the radiation's phase rather than energy, unlike electromagnetic radiation from astronomical sources. The coherent superposition of several wave cycles from a source creates the SNR. The signal structure and phase evolution provide more physics information than the amplitude. Tracking a signal's phase requires searching not only for individual sources but also throughout a large region of each source's parameter space, which puts a lot of strain on the theoretical understanding of waveforms and data analysis gear.

• Finally, gravitational wave detection requires calculation. Gravitational wave antennas collect data at many megabytes per second for years. Signal data makes up 1% of this data, whereas housekeeping data monitors detector operation. The huge parameter space above requires signal data to be filtered multiple times for different searches, which imposes heavy demands on computing power and algorithms.

Since the mid-1980s, wideband detector data analysis has advanced [64] [65]. Annual Gravitational Wave Data Analysis Workshops disseminate current thinking and challenges. These methods were originally tested with interferometers [51] and bars [15]. Although the theory is well understood [43], data analysis methodologies depend on computer resources, data volumes, astrophysical understanding, and source modeling and are constantly revised.

The data analysis method known as matched filtering effectively looks for a signal of known shape hidden in noisy data. The method entails comparing a waveform, often referred to as a template or filter, to the output of a detector. In the presence of noise $n(t)$ and a signal $h(t)$, the goal is to identify an "optimal" template $q(t)$ that results in the best SNR feasible on average. For more information, an interested reader might study any classic textbook on signal analysis, such as Helstrom [41].

We will refer to the detector output as $x(t)$, which is considered to be made up of a meaningful gravitational wave signal $h(t)$ and background noise $n(t)$. A quantity's Fourier transform, indicated by the symbol $\tilde{x}(f)$, is defined as

$$\tilde{x}(f) = \int_{-\infty}^{\infty} x(t) e^{2\pi i f t} \, \mathrm{d}t. \tag{2.10}$$

The detector output $x(t)$ is a realization of noise $n(t)$, i.e., $x(t) = n(t)$ when no signal is present. In the presence of a signal $h(t)$ with an arrival time t_a, $x(t)$ takes the form

$$x(t) = h\,(t - t_a) + n(t), \tag{2.11}$$

and the correlation c of a template $q(t)$ with the detector output is defined as

$$c(\tau) \equiv \int_{-\infty}^{\infty} x(t) q(t + \tau) \mathrm{d}t. \tag{2.12}$$

We will proceed to find the ideal filter $q(t)$ that, in the presence of a signal $h(t)$, optimizes the correlation $c(\tau)$. We begin by writing the correlation integral in the Fourier domain. To do this, let us first substitute the $\tilde{x}(f)$ and $\tilde{q}(f)$ values for the $x(t)$ and $q(t)$ values in the preceding integral. We have

$$c(\tau) = \int_{-\infty}^{\infty} \tilde{x}(f) \tilde{q}^*(f) e^{-2\pi i f \tau} \mathrm{d}f, \tag{2.13}$$

where $\tilde{q}^*(f)$ denotes the complex conjugate of $\tilde{q}(f)$.

In the same way that n is a random process, so is c. Additionally, since correlation is a linear process, it follows the same probability distribution function as n. In particular, if n has a Gaussian random process with zero mean, then c likewise has a Gaussian distribution function, though its mean and variance will typically be different from n. The mean value of c, denoted by $S \equiv \bar{c}$, is the correlation of the template q with the signal h, since the mean value of n is zero:

$$S \equiv \bar{c}(\tau) = \int_{-\infty}^{\infty} \tilde{h}(f)\tilde{q}^*(f)e^{-2\pi i f(\tau - t_a)}\mathrm{d}f. \tag{2.14}$$

The variance of c, denoted $N^2 \equiv \overline{(c-\bar{c})^2}$ is

$$N^2 = \overline{(c-\bar{c})^2} = \int_{-\infty}^{\infty} S_h(f)|\tilde{q}(f)|^2 \, \mathrm{d}f. \tag{2.15}$$

One then defines the SNR:= ρ through

$$\rho^2 \equiv S^2/N^2. \tag{2.16}$$

The last three equations above naturally lead to the concept of the waveforms' scalar product. Given two functions, $a(t)$ and $b(t)$, their scalar product $\langle a, b\rangle$ is defined as [29, 30, 21, 25]

$$\langle a, b\rangle \equiv 2\int_0^{\infty} \frac{\mathrm{d}f}{S_h(f)} \left[\tilde{a}(f)\tilde{b}^*(f) + \tilde{a}^*(f)\tilde{b}(f)\right]. \tag{2.17}$$

Also, we know that $S_h(f) \geq 0$ consequently, the scalar product is real and positive definite. Note that the Fourier transform of a real function $h(t)$ obeys $\tilde{h}(-f) = \tilde{h}^*(f)$.

Using the scalar product mentioned above, the SNR can be expressed as follows:

$$\rho^2 = \frac{\left\langle he^{2\pi i f(\tau - t_a)}, S_h q\right\rangle}{\sqrt{\langle S_h q, S_h q\rangle}}. \tag{2.18}$$

It is evident from this that the template q that obtains the maximum value of ρ is simply

$$\tilde{q}(f) = \gamma \frac{\tilde{h}(f)e^{i2\pi f(\tau - t_a)}}{S_h(f)}, \tag{2.19}$$

where γ is an arbitrary constant.

From the above equation, we know that the SNR is maximized when τ is equal to t_a and we also know that the noise power spectral density (PSD) $S_h(f)$, weighs down the ideal filter, which is not just a copy of the signal. Finally, one obtains the optimal SNR as

$$\rho_{\mathrm{opt}} = \langle h, h\rangle^{1/2} = 2\left[\int_0^{\infty} \mathrm{d}f \frac{|\tilde{h}(f)|^2}{S_h(f)}\right]^{1/2}. \tag{2.20}$$

Chapter 3

Artificial Neural Networks

3.1 Utilizing Artificial Neural Networks

Because of the large number of parameters that determine the emission of GW transients, the outcome of a parameter estimation search will heavily depend on the capability of the analysis to differentiate between GWs and glitches created by non-stationary and non-Gaussian noise tails. This difficulty will most likely continue to exist for burst signals even in the era of advanced detectors, even though it will be reduced by innovations in noise attenuation. Additionally, just like in the previous analyses, it will be particularly pronounced for poorly modeled searches. It is therefore a shared objective to lessen glitches' effect on GW search results. According to the information provided, the GW detection community has made multiple efforts to individuate the most distinctive characteristics of the signals about the glitches. Several of them are concerned with the implementation of various Machine Learning strategies. Recognition, regression, planning, prediction, and classification are machine learning algorithms' main tasks. These informatics structures rely heavily on learning by example. This introduces algorithms that are not fully determined, allowing for the consideration of relations not yet established [55].

3.1.1 Boosted Decision Trees

A boosted decision tree (BDT) is a collection of decision trees from the same training set with different weights, whose final judgment on an event is determined by the output with the most votes. Like Support Vector Machines (SVM), this algorithm distinguishes signal from noise. but the strategy is different. These devices have numerous nodes that consider one variable and apply the best value cut. The root node identifies the most different features from the two categories based on the whole training set. The algorithm separates events using a discriminant threshold. Each new subset gets a discriminant node after this training sample split. Next, a variable is chosen based on training set characteristics and a cut is applied. The technique is repeated until one of two conditions is met: a too-small subset for adding nodes or maximum/minimum signal purity.

Multiple decision trees are created from the same training set to better classify occurrences into two classes. This is done through boosting or bagging.

Boosting. Training events that are incorrectly classified receive heavier weights. In this way, the performance is enhanced by the many applications of this boosting procedure that create a forest of various tree sets.

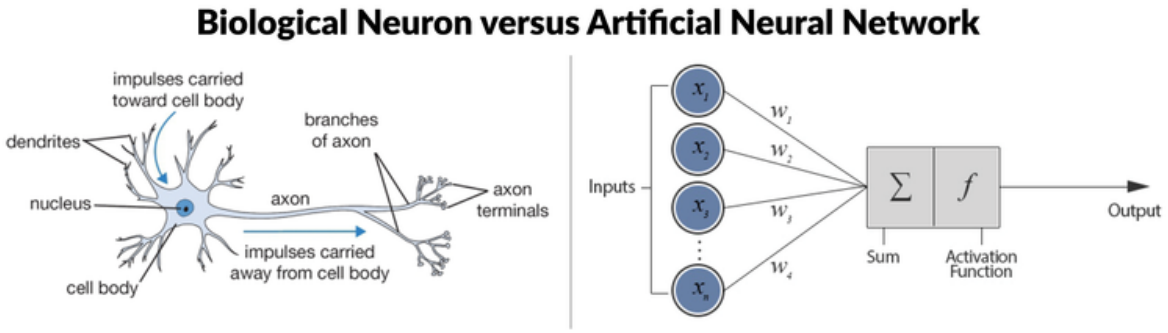

Figure 3.1: Biological Neuron versus Artificial Neural Network [70].

Bagging. The past performances are not taken into account when creating a decision tree forest. Based on a stochastic resampling (bootstrap) of the given training set, the set of decision trees is defined. A single decision tree is constructed using each of the new subsets. Event classification is the main function of this type of analysis. Some algorithms are testing BDTs to improve signal glitches discrimination.

3.1.2 Artificial Neural Networks (ANNs)

ANNs are algorithms based on brain function, this approach's generalism lets these informatic tools conduct regression, classification, pattern recognition, and more. The most common GW detection applications try to distinguish signal from noise. This challenge is usually solved by identifying glitches and classifying them ([10] [57] [12]). In this way, glitch identification is improved over typical deterministic studies. These studies also demonstrate that applying these models does not reduce signal detection and that glitches can be partially classified by noise origin [55].

3.1.3 Comparing Biological to Artificial Neural Networks

Neurons join each other in the biological brain network. Dendrites send inputs from other neurons to the cell nucleus, the soma. The soma calculates a weighted sum of dendrite electrical impulses. The amount of information transported relies on numerous factors: input signal strength, dragged link importance, weights, and most importantly, the neuron's activation threshold. Soma data analysis determines passive and active behavior. The soma does not generate impulses but produces spikes when inputs reach a certain weighted sum. The axon sends the output result to the synapses connecting different nerve cells. Electrical impulses release neurotransmitters, which carry information from the presynaptic neuron's axon to its dendrites. Post-synaptic processes link the chemical substance to electrical impulses. Artificial neural networks are informatic algorithms. Neurons calculate the weighted total of inputs using an activation function. These systems are fascinating because they can learn by example. Changing synaptic weights optimizes learning. Network topology and neuron characterization define an artificial neural network ready to evaluate incoming signals.

In Figure 3.1 we can see how similar is the biological neuron to the artificial neuron. For classification and prediction tasks, artificial neural networks (ANNs) are an excellent replacement for traditional multivariate statistical methods. Software-based artificial neural

networks (ANNs) mimic biological neural networks in that they are built of the same basic processing units—neurons—that are connected to the outside world and/or to other neurons. The multilayer perceptron (MLP), one of the most popular network topologies, has three or more layers of neurons: input neurons, which take in and process discrete or continuous inputs from the data set; output neurons, which give the result back; and one or more hidden neuron layers. Synapses connect the artificial neurons, which behave like biological neurons in that they take a (weighted) input from the outside world or other neurons, and then use a transfer or activation function to process the total of the inputs and either send it to other neurons or produce outcomes [61].

The network structure is primarily influenced by the strength of the synapses (weights), the type of connections, and the number of neurons in the network, among neurons. The feedforward and recursive neural networks are two different types of networks that can be distinguished based on the directions in which data propagates. While in the latter situation, the architecture is characterized by the presence of certain feedback connections between the neurons, which consequently need to be dynamic, the structure in the former can be represented by an oriented graph. The basic goal of these algorithms is to identify an informatic structure that can accurately understand the incoming data using a series of samples termed a training set. The ability of these devices to generalize learning to samples outside the training set is a crucial quality to assess. The learning processes train the neural networks by changing the connection weights, simulating the natural process that alters the synapsis thickness over time, and weakening or strengthening the connections between neurons. The first category is Unsupervised learning, when the output neurons from the training set are not available or are not denied, and supervised learning, where the network parameters are defined by using clustering techniques on the input training sample. The second category is supervised learning where an algorithm that minimizes the error (or cost) function on the training set is used to determine the weights while accounting for the network's desired output.

There can be different choices for the cost functions and two examples are shown

$$\begin{aligned} E(w) &= \frac{1}{2}\sum_{p=1}^{P}\left(\left\|y\left(x^p, w\right) - t^p\right\|^2\right), \\ E(w) &= -\sum_{p=1}^{P}\left(t^p \log\left[y\left(x^p, w\right)\right] + \left(1 - t^p\right)\log\left[1 - y\left(x^p, w\right)\right]\right), \end{aligned} \tag{3.1}$$

where y is the output of the ANN, the label p refers to the training set made up of P samples, t is the target vector, or the vector that the network should produce, and w and x are the input vectors' respective weights in the network. The acquisition or use of the training set over time is another difference between the learning procedures [40] [55].

3.1.4 Perceptron and Multilayer Perceptron

The digital equivalent of a biological neuron is a perceptron. The following provides examples of a perceptron's key characteristics; in particular, we concentrate on its capacity to linearly classify the data by solving, under a set of predetermined criteria, a linear inequality system. Neurons are the basic computational units that make up a neural network. In 1943, McCulloch and Pitts [31] suggested one of the most basic structures for this algorithm, which was further refined by Rosemblatt [59]. The neuron elaborates a vector of input values in this model by multiplying them by weights, computing the weighted total, and then comparing it to a

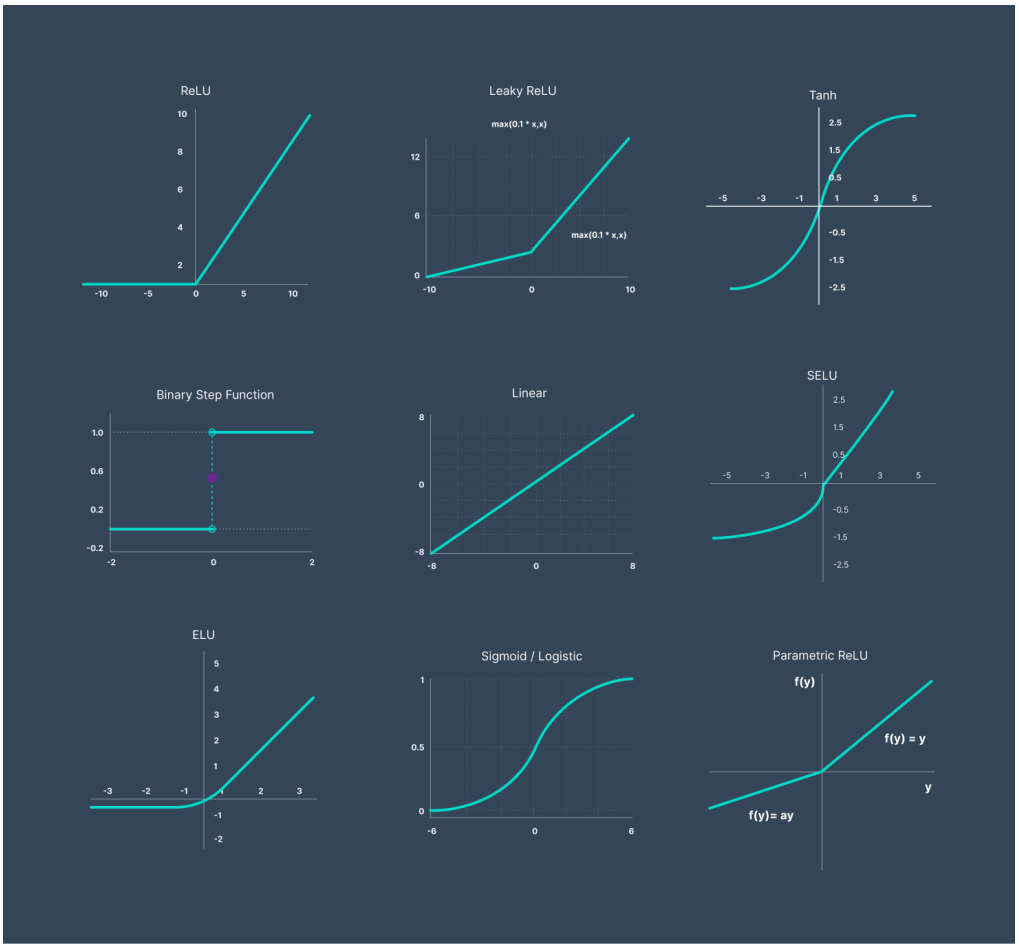

Figure 3.2: Some activation functions that are commonly in use [43].

threshold value to yield the scalar response. If the sum is higher than the threshold, the neuron outputs 1, otherwise, it outputs 0.

Formally, the neuron operation can be defined as follows. The following equation gives the neuron output y for an input vector $x \in \mathbb{R}^n$, a corresponding weight vector $w \in \mathbb{R}^n$, and a threshold θ:

$$y(x) = g\left(\sum_{i=1}^{N} w_i x_i - \theta\right) \equiv g\left(w^T x - \theta\right), \tag{3.2}$$

where g is the activation function of the neuron, and in this particular instance, it can be characterized by the sign function:

$$g(t) \equiv \operatorname{sgn}(t) = \begin{cases} 1, & \text{if} \quad t \geq 0 \\ -1, & \text{if} \quad t < 0 \end{cases}. \tag{3.3}$$

In a space with just two dimensions, the sign-evaluation of a weighted sum is determined by an inequality that is constrained by the implicit equation of a straight line

$$\sum_{i=0}^{n} w_i x_i - \theta = w_1 x_1 + w_2 x_2 - \theta > 0. \tag{3.4}$$

Figure 3.2 shows some of the most used activation factions. Many are are variations of RELU (Rectified Linear Unit).

The following are the limitations of a Perceptron model. First is that the output of a perceptron can only be a binary number (0 or 1) due to the hard-edge transfer function. Second is that it can only be used to classify the linearly separable sets of input vectors. If the input vectors are non-linear, it is not easy to classify them correctly [55].

Multilayer perceptrons are used in a subset of ANNs. The neurons in these devices are arranged in layers of structures. Here, we emphasize feedforward networks made up of trained layers using supervised learning techniques and static neurons. We are interested in function approximation; in this field, approximation theory governs ANN performance. The choice of the function class used for the approximation job, which typically depends on the weights in a non-linear fashion, and the consideration of a discrete series of data in the structure definition are two characteristics that define ANNs. These characteristics make the usage of ANNs very beneficial, such as their capacity to build a relationship between the input-output values that have not yet been discovered, but they also present some issues and areas that require further development. Starting with the discretization of the data, there are an infinite number of functions that can accurately understand the input-output pairs of the training set. The potential for noise in the data, which may affect them and the resulting network parameters, is another problem related to them. The characterization of the approximation degree about the input number, the ANN structure, and other approximation aspects are the final issues that come up. However, a conclusion appears to be obvious: multilayer perceptrons are universal approximators, and an ANN with a finite number of hidden layers can approximate any limited function with arbitrary precision. The solution to the learning problem is to identify the optimal method for describing the training data and to develop an algorithm capable of identifying the fundamental characteristics of the inputs. As a result, a delicate balance must be struck between the ability to generalize and the proper interpretation of training data: too simple models do not capture coarse linkages between inputs and outputs, while too complicated models can be confined to the proper development of the single training set [55].

3.1.5 Multilayer Perceptron Structure

A perceptron is a useful tool for doing classical separation of a subset of events that may be separated linearly. The perception's applicability is severely constrained by this subsets conjecture; for instance, it cannot carry out the XOR logic operation. A collection of functions that specify how to change the data in a space where they are now linearly separable can be introduced to alleviate this issue. This strategy was initially put forth by Rosenblatt, however, it has several drawbacks because the functions that can be employed are limited. The perceptron limits have encouraged the development of new tools targeted at solving more challenging problems, such as the multilayer perceptron, which is created by starting with a set of nonlinearly separable perceptrons and dividing them into ordered structures.

A collection of neurons arranged into different classes and referred to as layers define the architecture of multilayer perceptrons:

• **Input layer**: Any calculation is done because the input layer is made up of n nodes connected to n inputs and has a transfer function of 1. This class's objective is to weigh each input and amount differently for the subsequent layer's perceptions.

• **Hidden layers**: The analysis of the data begins with the hidden layers. They are made up of computational units arranged in several succeeding layers. The goal of the neurons in the hidden layers is the development of the output-input couples so that different relationships between the inputs and their functions may be discovered and tested.

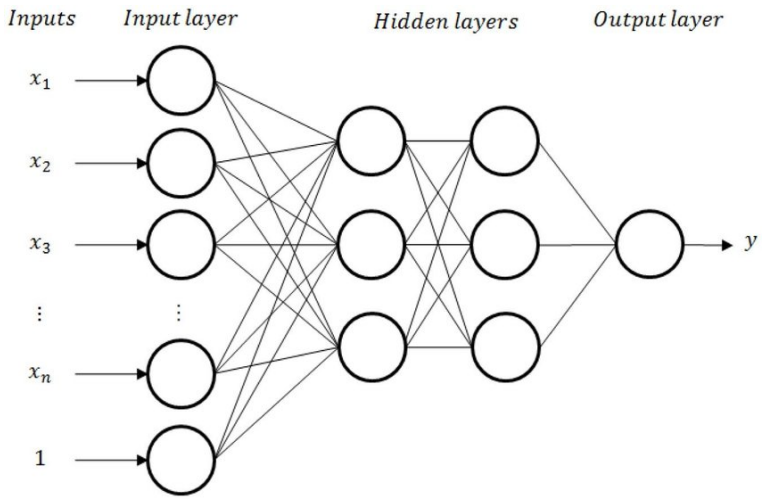

Figure 3.3: Multilayer perceptron [13].

• **Output layer**: The output layer is the neuron class that returns the outputs of the network. They use the weighted sums of the outputs from the last hidden layer to make the final output calculations.

Figure 3.3 shows the architecture of a multilayer perceptron with 2 hidden layers.

This particular type of neural network exhibits a specific type of synapse: the connections only occur between neurons in two successive layers. Information goes in a single path and relationships between neurons from the same class are not taken into account (no feedback connections). The calculation unit j from the layer $l = 1, ..., L$ can be used to identify the connection and the neuron or node i of the previous layer. To determine L, we have taken into account a subclass of $L \geq 2$ layers that only contains the hidden and output neurons. A characteristic of these neurons is their activation function $g_j^{(l)} : \mathbb{R} \mapsto \mathbb{R}$ used to evaluate the inputs' weighted sum. As a result, the computations are as follows:

$$a_j^{(1)} = \sum_{i=0}^{n} w_{ji}^{(1)} x_i, \quad z_j^{(1)} = g_j^{(1)}\left(a_j^{(1)}\right), \quad w_{j0} = \theta_j, \quad x_0 = -1, \quad l = 1, \tag{3.5}$$

$$a_j^{(l)} = \sum_{j}^{N^{(l-1)}} w_{ji}^{(l)} z_i^{(l-1)}, \quad z_j^{(l)} = g_j^{(1)}\left(a_j^{(l)}\right), \quad w_{j0} = \theta_j, \quad z_0^{(l-1)} = -1, \quad l > 1, \tag{3.6}$$

where we have introduced the number of neurons in the l layer N^l. Typically, algorithms of this type have differentiable activation functions with sinusoidal shapes, like logistic function and hyperbolic tangent. Whatever the case, different neurons can have different activation functions (typically, the output layer can have a different activation function compared to the hidden layers). Some additional definitions for the notation used above are:
• number of neurons in the hidden layer: N
• threshold of the hidden neuron j: θ_j
• weights between the inputs and the first hidden layer: w_{ij}
• weights between the hidden layer and the output: v_j

The output is produced by the equation below when an activation function, linear for the output neuron and g for the hidden perceptrons, is considered

$$y(x) = \sum_{j=1}^{N} v_j g\left(\sum_{i=1}^{n} w_{ji} x_i - \theta_j\right) = \sum_{j=1}^{N} v_j g\left(w_j^T x - \theta_j\right). \tag{3.7}$$

A neural network with a single hidden layer approximates continuous functions on compact sets of $\mathbb{R}^n$ over a wide range of activation functions, particularly for non-polynomial continuous functions on $\mathbb{R}$ [53]. Formally this result states that: given a function $f(x) \in C(\mathbb{R}^n)$, a compact $\Omega \in \mathbb{R}^n$ and a $\varepsilon > 0$ we can build a multilayer perceptron with a single hidden layer, characterized by a continuous nonpolynomial activation function, which satisfies

$$\max_{x \in \Omega} |f(x) - y(x)| < \varepsilon. \tag{3.8}$$

The theory approximation for MLPs with more than one hidden layer is not well known, making it difficult to assess the benefits and drawbacks that this structure offers in comparison to the MLPs with just one hidden layer. The degree of approximation is indeed constrained to a lower bound for architectures with a single hidden layer and depends on the number of neurons employed. In practice, neural networks with multiple hidden layers have proven to be very effective.

The design of an MLP with n inputs and K outputs, $L+1$ layers and $N^{(l)}$ $(l = 1, \dots L)$ calculation units for each layer, as well as the learning method affect the MLP's capacity for data interpretation. Both are crucial for the accurate assessment of the weight vectors; specifically, the former establishes the quantities of these parameters, w_{ij}^l, while the latter addresses the algorithm's capacity to arrive at ideal values. The size of the training set, or the number of instances used to calculate the weights, must also be considered while selecting these parameters:

$$T = \left\{\left(x^p, t^p\right), \quad x^p \in \mathbb{R}^n, \quad t^p \in \mathbb{R}^K, p = 1, \dots P\right\}. \tag{3.9}$$

The approach used to determine the weight vectors aims to find the minimum of the error function, also known as the objective function. In our situation, this entails solving an optimization problems:

$$\min_{w \in \mathbb{R}^n} E(w) = \sum_{p=1}^{P} E_p(w) = \sum_{p=1}^{P} \frac{1}{2} \left\|y^p - t^p\right\|^2. \tag{3.10}$$

Some theories have been established to define the architecture to determine the bare minimum of events required for the accurate definition of the weights. They are frequently limited to specific instances, and different strategies are used in actual practice. The fundamental principle underlying these methods is to train the algorithm after evaluating the architecture using various sets of events. Test structures with an increasing or decreasing number of perceptrons are used to determine the number of neurons for each layer. When testing structures with only one hidden layer, this approach is reasonable to use, but when testing structures with multiple hidden layers, this procedure is computationally expensive and time-consuming.

3.1.6 Back Propagation

By providing the network with input-output pairs, it can learn how to classify inputs correctly. The synapses are initially uniformly distributed but randomly weighted and they are updated throughout training to minimize the objective function. Various tactics can be used to achieve the optimum configuration. These methods, also known as learning strategies,

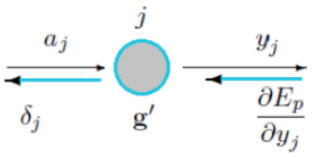

Figure 3.4: The image displays information propagation when backpropagation is applied and the examined j-neuron is a part of the output layer [55].

typically include at least the first derivative of the error function concerning the weights. The back-propagation (BP) algorithm is frequently used to compute derivatives. There are two distinct versions available:

• **Mini-batch BP** The weights are changed after taking into account a certain number of training examples.

• **Stochastic BP** the weights are adjusted for every example in the training.

We are thinking about an MLP with a L layer, where the input vector is $x \in \mathbb{R}^n$ and the output vector is $y \in \mathbb{R}^K$. We denote the vectors z to make the notation simpler.

$$z_i^{(0)} = x_i, \quad i = 1, \dots n \quad z_i^{(L)} = y_i, \quad i = 1, \dots K. \tag{3.11}$$

The derivative rules for composite functions are used in the back-propagation process

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \tag{3.12}$$

Denoting $\delta_j = \frac{\partial E_p}{\partial a_j}$ it is calculated with two distinct instances in mind:

• the j-neuron belongs to the output layer

$$\delta_j \equiv \frac{\partial E_p}{\partial a_j} = g\left(a_j\right)' \frac{\partial E_p}{\partial y_j}, \tag{3.13}$$

where $y_j = g\left(a_j\right)$.

• The j-neuron is a part of a hidden layer

$$\delta_j \equiv \frac{\partial E_p}{\partial a_j} = \sum_k \frac{\partial E_p}{\partial a_k} \frac{\partial a_k}{\partial a_j}, \tag{3.14}$$

where $\partial a_k / \partial a_j = g\left(a_j\right)' w_{kj}$.

Backpropagation is the name given to the calculation of this second term because of the utility of neuron output. Therefore:

$$\frac{\partial E_p}{\partial w_{ji}} = \delta_j z_i. \tag{3.15}$$

As a result, using backpropagation, we can determine the gradient by adding the gradients of all the training examples. Therefore, the computing cost of evaluating the whole gradient is $O(P \times W)$, where P are the total number of events in the training set and W is the total number of weights. The direction of the search can be changed to speed up convergence. This is accomplished by adding a new term to the weight-iterative formula proportional to the difference between the weight values of the previous two updates. The momentum method is the name of the strategy used to hasten the accomplishment of the error function minimum.

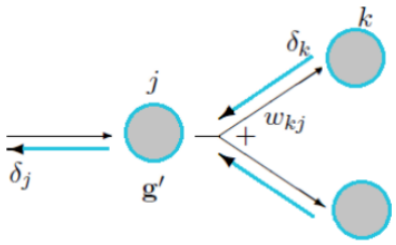

Figure 3.5: The propagation of information when backpropagation is applied and the examined j-neuron is a member of a hidden layer [55].

3.1.7 Gradient methods

Various tactics have been used to carry out the training process. Some of them demand monotonicity in the evaluation of the error function about each weight update. However, they frequently have computational drawbacks, leading to the recent introduction of a novel method. It is based on an iterative procedure that permits a temporary increase in the error function while still guaranteeing the same global convergence quality. Various, potential advancements of this approach are available.

Let's begin with the first class of suggested learning methods: gradient techniques. The subsequent iterative process for updating the weight results in one of the simplest versions:

$$w^{k+1} = w^k - \eta^k \nabla E\left(w^k\right). \tag{3.16}$$

Each iteration in this case is identified by the index k and relates to an epoch. Convergence is guaranteed if the learning rate is assumed to remain constant during the iteration $\eta^k = \eta, \quad \forall k$, subject to certain restrictions on the error function. Regardless, the procedure for determining the parameters required for the estimation might be difficult, so a different technique is typically used. It has to do with choosing a learning rate whose value is determined by the iteration index k. The different methods use the values provided by η^k to effectively move and reduce the objective function.

The fundamental goal of these methods is to minimize the error function by moving in the opposite direction as the gradient, however as previously stated, overall performance greatly depends on the learning rate. The algorithm can overcome some of its challenges with this parameter's help when given the right values. For instance, it can be helpful to escape from local minima. The Robbins-Monro stochastic minimization, which is based on the gradient study, is the easiest learning technique used with the ROOT toolkit (TMVA) [67]. It is an stochastic approach in a modified form, and the following provides the updating rule:

$$\Delta w_{ij}^k = -\eta \left(\frac{\partial E_p}{\partial w_{ij}} + \delta\right) + \epsilon \Delta w_{ij}^{k-1}. \tag{3.17}$$

Therefore, the gradient is used to search for the error function minimum, but an additional element is added for at-spot elimination. Additionally, the technique takes into account the first history of the progression of weight using the term Δw_{ij}^{k-1}. We can see that three parameters must be determined to apply this algorithm. ROOT, however, suggests several acceptable values for these parameters.

3.1.8 Optimizers

Optimizers play a crucial role in the training of machine learning models. The goal of an optimizer is to find the best set of parameters for the model that minimizes the loss function. This is achieved by updating the parameters based on the gradients of the loss function with respect to the model parameters. In other words, the optimizer is responsible for adjusting the parameters in such a way that the model's prediction error is reduced. The loss function is a measure of how well the model fits the training data, and the gradients of the loss function represent the direction of the steepest increase in the loss. The optimizer takes the gradients into account and updates the parameters accordingly, effectively "stepping" towards a lower loss. This process is repeated multiple times until the optimizer converges to a minimum in the loss landscape. There are various optimization algorithms available, including first-order methods and second-order methods. First-order methods, such as gradient descent, update the parameters based on the gradient of the loss with respect to the parameters. Second-order methods, such as Newton's method, take into account the curvature of the loss surface and update the parameters accordingly. In general, first-order methods are more computationally efficient and well-suited for large-scale problems, while second-order methods are more accurate but more computationally expensive. There are also hybrid methods that combine the strengths of first-order and second-order methods. It is important to note that the optimization process is an iterative one, and the choice of optimizer can greatly affect the convergence speed and the final solution. In practice, researchers often try several different optimizers and choose the one that works best for their specific problem. In conclusion, optimizers are an essential component of machine learning and play a crucial role in the training of models. The choice of optimizer can greatly impact the performance of a model and researchers should carefully consider the strengths and weaknesses of different optimization algorithms when selecting an optimizer [36] [19]. Some examples of optimizers are:

• **ADAM**. Adaptive Moment Estimation was introduced by Kingma and Ba in 2014. Adam is a first-order gradient-based optimization method that works by iteratively updating the parameters based on the gradient of the loss function. The key idea behind Adam is to use moving averages of the gradients and squared gradients to dynamically adjust the learning rate for each parameter. This helps to reduce oscillations and speed up convergence, which are common problems in gradient-based optimization.

The update rule for Adam is given by:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla_\theta J(\theta),$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \nabla_\theta J(\theta)^2,$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t},$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t},$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

Here, m_t and v_t are the moving averages of the gradients and squared gradients, respectively; $\hat{m}_t$ and $\hat{v}_t$ are the biased estimates of the moving averages; θ_t is the updated parameter; β_1

and β_2 are the decay rates for the moving averages; α is the learning rate; and ϵ is a small constant to prevent division by zero.

The key advantage of Adam is its ability to adjust the learning rate for each parameter dynamically. This helps to ensure that the model is making progress towards a minimum of the loss function, even if the gradients are not consistently pointing in the same direction. This is particularly important for complex models, where the loss function can have many local minima. However, it is important to note that Adam requires tuning of two hyperparameters: the learning rate and the decay rates for the moving averages (β_1 and β_2). These hyperparameters need to be set carefully to ensure that the model is able to converge efficiently. In practice, the default values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ work well for most applications.

In conclusion, Adam is a powerful optimization algorithm that has become a popular choice for training machine learning models. Its combination of moving averages of the gradients and squared gradients, along with a dynamic adjustment of the learning rate, provides a framework for making efficient updates to the parameters of the model [45].

• **NAdam**. NAdam, which stands for Nesterov-Accelerated Adaptive Moment Estimation, is a variant of the Adam optimization algorithm that incorporates ideas from Nesterov's Accelerated Gradient (NAG) method. The main idea behind NAdam is to incorporate the NAG approach into the Adam optimization algorithm, which can further improve the convergence speed of the model.

The NAdam update rule is given by:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla_\theta J(\theta),$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \nabla_\theta J(\theta)^2,$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t},$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t},$$

$$\theta_t = \theta_{t-1} + \beta_1 \cdot (\hat{m}t - mt - 1) + (1 - \beta_1) \cdot \nabla_\theta J(\theta),$$

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

Here, m_t and v_t are the moving averages of the gradients and squared gradients, respectively; $\hat{m}_t$ and $\hat{v}_t$ are the biased estimates of the moving averages; θ_t and θ_{t+1} are the updated parameters; β_1 and β_2 are the decay rates for the moving averages; α is the learning rate; and ϵ is a small constant to prevent division by zero.

The main difference between NAdam and Adam is that NAdam incorporates the idea of NAG into the update rule, which helps to prevent oscillations and speed up convergence. This is achieved by computing the gradient at a point in the future, and then using that gradient in the update rule. This helps to ensure that the model is moving in the direction of the minimum of the loss function, even if the gradient is not pointing directly at the minimum. Like Adam, NAdam requires tuning of the hyperparameters, including the learning rate and the decay rates for the moving averages. In practice, NAdam can provide faster convergence and better results compared to Adam, especially for complex models with many local minima in the loss function.

In conclusion, NAdam is a powerful optimization algorithm that builds on the ideas of Adam and NAG to provide a more efficient and effective method for updating the parameters of a machine learning model [60].

• **RMSProp**. Root Mean Square Propagation (RMSProp) is an optimization algorithm that is widely used in deep learning. It is a variation of gradient descent that uses moving averages to normalize the gradient updates, making the optimization process more stable and efficient. The RMSProp update rule is given by:

$$s_t = \beta \cdot s_{t-1} + (1 - \beta) \cdot \nabla_\theta J(\theta)^2,$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{s_t + \epsilon}} \cdot \nabla_\theta J(\theta).$$

Here, s_t is the moving average of the squared gradients; θ_t and θ_{t+1} are the updated parameters; β is the decay rate for the moving average; α is the learning rate; and ϵ is a small constant to prevent division by zero.

The main idea behind RMSProp is to normalize the gradient updates by dividing the gradient by the root mean square of the past squared gradients. This helps to prevent the optimization process from being dominated by the largest gradients, which can cause the model to oscillate and converge slowly. The decay rate β controls how much weight is given to the previous squared gradients, with a lower value of β giving more weight to the most recent gradients. In practice, RMSProp is a robust optimization algorithm that has been found to work well for deep learning models, especially for models with large numbers of parameters and highly non-linear functions. It is often used in combination with other optimization algorithms, such as Momentum or Adam, to further improve the stability and convergence speed of the optimization process.

In conclusion, RMSProp is a widely used optimization algorithm that provides a powerful and efficient method for updating the parameters of a machine learning model. It is particularly useful for deep learning models with large numbers of parameters, where it can provide faster convergence and better results compared to standard gradient descent [68].

• **Adamax**. Adamax is an optimization algorithm for deep learning models that is based on the Adam optimization algorithm. Adamax is a variant of the Adam optimizer that uses a different formula for computing adaptive learning rates for each parameter.

The Adamax optimization algorithm updates the parameters of the model based on the gradients of the loss function with respect to the parameters. Like other variants of Adam, Adamax uses a moving average of the historical gradients to compute the adaptive learning rates. However, unlike Adam, Adamax uses the infinity norm of the gradients, rather than their squared L_2 norm, to control the learning rate. This means that the learning rate for each parameter is scaled based on the maximum gradient magnitude, rather than the average gradient magnitude. In mathematical terms, the Adamax optimization algorithm can be expressed as follows. Let w_t be the parameters of the model at time step t, g_t the gradient of the loss function with respect to the parameters at time step t, m_t and v_t the moving averages of the gradients and their squares, respectively, β_1 the decay rate for the first moment, β_2 the decay rate for the second moment, and ϵ a small positive constant. Then, at each time step t, the parameters are updated as follows:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t,$$

$$v_t = \max(\beta_2 \cdot v_{t-1}, |g_t|),$$

$$w_t = w_{t-1} - \alpha \cdot \frac{m_t}{v_t + \epsilon}.$$

Here, α is the learning rate, which can be set by the user or determined through a hyperparameter optimization process.

The Adamax optimization algorithm has been shown to perform well in a variety of deep learning tasks, including image classification, natural language processing, and reinforcement learning. Adamax is particularly well suited to tasks where the gradients can have very large magnitudes, as it is able to handle these large gradients effectively by scaling the learning rate accordingly [45].

• **AMSGrad**. Adaptive Moment Estimation Gradient (AMSGrad) is a variant of the Adam optimization algorithm, which is widely used in deep learning and machine learning. AMSGrad was introduced in 2018 by Reddi et al. in a paper entitled "On the Convergence of Adam and Beyond".

AMSGrad modifies the update rule of the Adam algorithm to improve its convergence properties, particularly in the case of non-convex optimization problems. In Adam, the learning rates are dynamically adjusted based on the exponential moving averages of the first and second moments of the gradients, which are used to approximate the mean and variance of the gradients. In AMSGrad, a maximum learning rate is introduced, which is used to clip the learning rates, ensuring that they do not become too large. Mathematically, AMSGrad can be described as follows: Let m_t and v_t denote the exponential moving averages of the first and second moments of the gradients, respectively. The parameters of the model are updated as follows:

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t,$$

$$v_t \leftarrow \max(\beta_2 v_{t-1}, |g_t|),$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{v_t} + \epsilon} m_t,$$

where g_t is the gradient of the loss function with respect to the parameters at time step t, α is the learning rate, β_1 and β_2 are the decay rates for the moving averages, ϵ is a small positive constant used to prevent division by zero, and θ_t denotes the parameters at time step t.

AMSGrad has been shown to be effective in a range of optimization problems, particularly in the case of deep learning models with large numbers of parameters. It has been used in a number of state-of-the-art models, and has been found to be particularly effective in tasks such as image classification, natural language processing, and reinforcement learning [58].

• **Adagrad**. Adagrad (Adaptive Gradient Algorithm) is a popular optimization algorithm that was introduced by John Duchi et al. in 2011. Adagrad adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters.

The core idea behind Adagrad is to maintain a per-parameter learning rate that is updated over time based on the historical gradient information. The learning rate is adjusted dynamically such that larger updates are performed for infrequent parameters, and smaller updates are performed for frequent parameters. Mathematically, Adagrad can be described

as follows: Let g_t be the gradient of the loss function concerning the parameters at time step t, and let G_t be the sum of squares of the gradients accumulated so far:

$$G_t \leftarrow G_{t-1} + g_t^2,$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t} + \epsilon} g_t,$$

where α is the initial learning rate, ϵ is a small positive constant used to prevent division by zero, and θ_t denotes the parameters at time step t.

Adagrad has been widely used in a variety of optimization problems, including linear regression, logistic regression, and neural networks. However, one of its main limitations is that the learning rate is constantly shrinking over time, making it difficult to escape from poor local minima. This issue can be addressed by using a larger initial learning rate or by gradually reducing the learning rate over time [28].

• **Adadelta**. Adadelta is an optimization algorithm that was introduced by Matthew Zeiler in 2012. It is an extension of Adagrad that addresses the issue of monotonically decreasing learning rates by introducing the concept of an accumulated moving average.

Like Adagrad, Adadelta uses per-parameter learning rates that are updated based on historical gradient information. However, instead of summing the squares of the gradients, Adadelta computes a moving average of the squared gradient updates. This moving average is used to determine the per-parameter learning rates, which are then used to update the parameters. Mathematically, Adadelta can be described as follows: Let g_t be the gradient of the loss function concerning the parameters at time step t, and let $E[g^2]_t$ be the moving average of the squared gradients:

$$E[g^2]t = \rho E[g^2]t - 1 + (1 - \rho) g_t^2,$$

$$\Delta\theta_t = -\frac{\sqrt{E[\Delta\theta^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} g_t,$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t,$$

where ρ is a decay rate, ϵ is a small positive constant used to prevent division by zero, and θ_t denotes the parameters at time step t.

Adadelta is a robust optimization algorithm that is well-suited for problems with noisy gradients or changing distributions. It has been shown to perform well on a variety of tasks, including training deep neural networks [72].

Chapter 4

Reference Model

This study utilizes a dataset with 1 million records pertaining to the Signal-to-Noise Ratio (SNR) of a population of binary black hole mergers. The black holes have random spins and they have masses between $7M_{\odot}$ and $50M_{\odot}$. They are randomly placed in terms of sky localization, spin orientation and inclination the binary plane.The distance is fixed to 1 Mpc (megaparsec). Since the SNR is inversely proportional to distance, our results can easily be scaled to any other distance of the source.

First, the target variable and predictor variables are separated, with the target variable "SNR" and the predictors being a list of 11 features. The values of these variables are then stored in X and y. After standardizing the data, it is then split into a training set and a test set, with 70% of the data being used for training and the remaining 30% for testing. We implement an artificial neural network (ANN) using the Keras library in python. The number of neurons in each layer of the network is initially set to 250. The network has initially 7 hidden layers and one output layer, all of which use the dense connection type. The activation function for each hidden layer is PReLU (parametric rectified linear unit) with a constant alpha initializer of 0.25. The model is then compiled with the 'mean squared error' loss function and the Nadam optimizer. The model is trained on the training data initially for 400 epochs with a batch size of 1000.

The Mean Absolute Percentage Error (MAPE) is a widely used metric for evaluating the accuracy of a model's predictions. It measures the average absolute percentage difference between the predicted values and the actual values. MAPE is a useful tool in the evaluation of a model's performance because it provides a measure of the magnitude of the prediction errors in percentage terms and is defined as:

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|actual_i - predicted_i|}{|actual_i|},$$

where n is the number of data points $actual_i$ is the actual value of the i-th data point and $predicted_i$ is the corresponding predicted value of the i-th data point It is important to note that MAPE is undefined for actual values equal to zero. This can cause issues when working with datasets that contain zeros. In such cases, alternative error metrics should be considered. In conclusion, the MAPE provides a useful measure of the accuracy of a model's predictions and can be helpful in comparing models or identifying high-error predictions. Its interpretation as a percentage error makes it an intuitive and easily understandable metric.

We also define

$$Accuracy = 100 - MAPE.$$

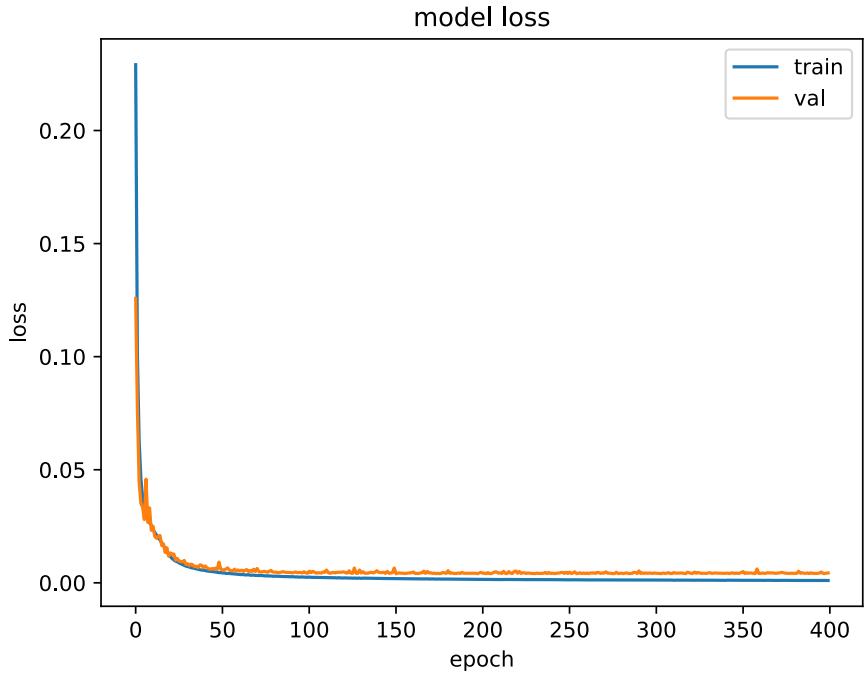

Figure 4.1: Loss functions in terms of epochs for a machine learning model during training and validation. The blue line represents the training loss and the orange line represents the validation loss.

For the above setting, we obtain $MAPE = 3.96$ and $accuracy = 96.04\%$.

In Figure 4.1 we display the change in the loss of a machine learning model over the course of its training. The blue line represents the loss on the training data, and the orange line represents the loss on validation data. The x-axis represents the number of epochs (iterations over the entire training data), and the y-axis represents the value of the loss. The loss is a measure of how well the model is fitting the training data, with a lower loss indicating a better fit. We observe how the loss decreases over the first few epochs, and then it may level out or continue to decrease gradually.

Figure 4.2 shows a histogram of the errors (true value minus predicted value) for the test data set. A few outliers are at errors up to ±6, but the majority are confined to less than ±2. This information can be useful in further refining and improving the model.

Figure 4.3 shows a histogram of the actual SNR values (blue) and the predicted SNR values (orange). By comparing the two histograms, one can assess the accuracy of the model's predictions and identify any differences between the two distributions. For example, if the model's predictions closely match the original testing data, the two histograms should be similar in shape and distribution. On the other hand, if the model's predictions are significantly different from the original testing data, the two histograms will likely be noticeably different. In this example, we observe a very good qualitative and quantitative agreement between actual and predicted distributions.

Finally, Figure 4.4 shows a scatter plot that compares the predicted Signal-to-Noise Ratio (SNR) values versus the actural SNR values for the test dataset. Each point on the plot represents a single prediction made by the model. The plot provides a visual representation of the accuracy of the model's predictions. We notice that there is a compact distribution of predicted values along the diagonal line, which corresponds to the actual values.

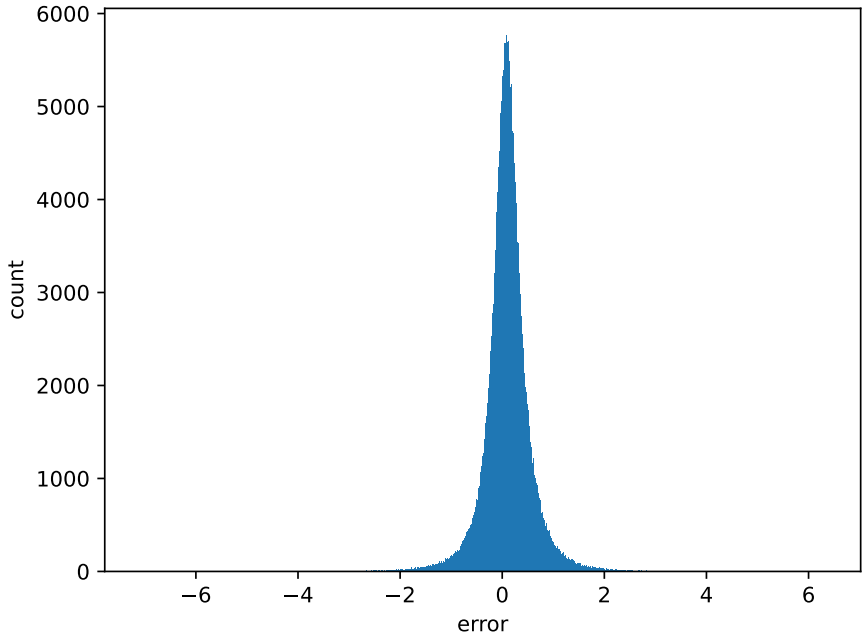

Figure 4.2: Histogram of the errors (true value minus predicted value) for the test data set. A few outliers are at errors up to ±6, but the majority are confined to less than ±2.

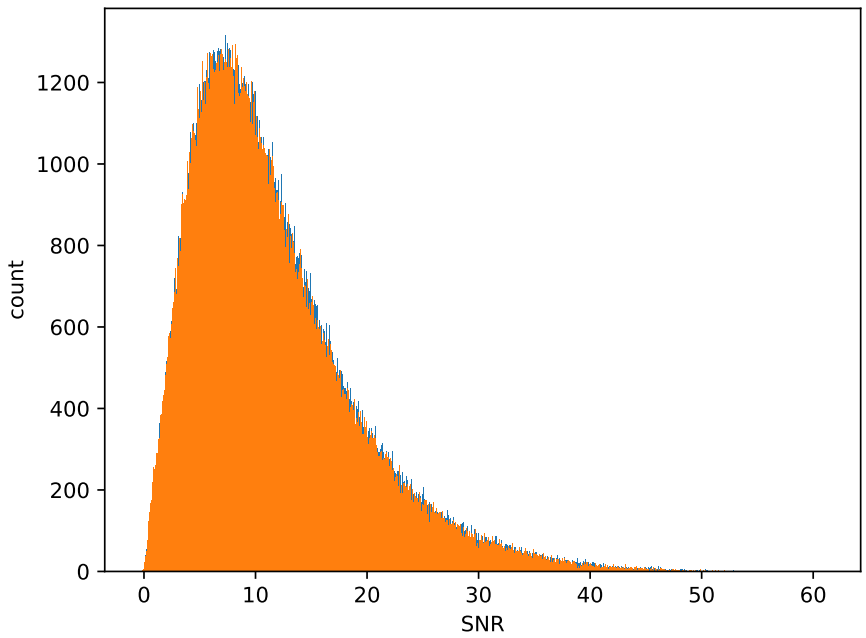

Figure 4.3: Histogram of the actual SNR values (blue) and the predicted SNR values (orange). We observe a very good qualitative and quantitative agreement.

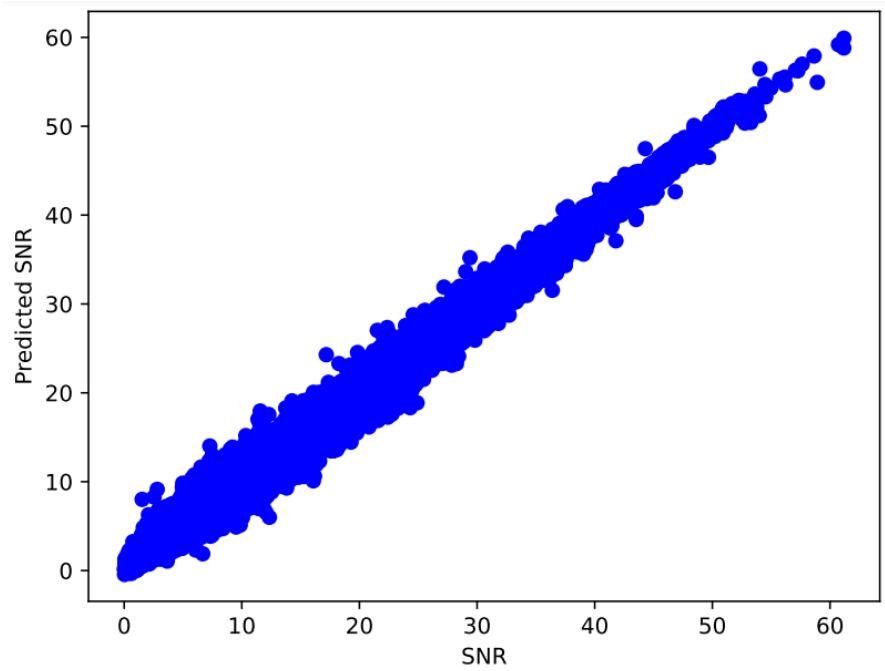

Figure 4.4: Scatter plot comparing the predicted Signal-to-Noise Ratio (SNR) values versus the actural SNR values for the test dataset. Each point on the plot represents a single prediction made by the model. The plot provides a visual representation of the accuracy of the model's predictions.

Chapter 5

Hyperparameter Optimization

Hyperparameters are variables in a machine learning model that are set before training and influence the model's behavior and performance. They determine the configuration of the model and are often critical in achieving optimal performance. Common examples of hyperparameters include learning rate, batch size, and a number of epochs. The selection of appropriate values for hyperparameters is crucial for training an effective machine learning model. In this section, we will be focusing on the optimization of two important hyperparameters, batch size and the number of epochs, through a method known as grid search [22].

• **Batch size**: Batch size is an important hyperparameter in the training of a machine learning model. The batch size determines the number of samples that will be processed by the model at once, before the weights are updated. In general, a big batch size means that the model will process more samples at once, which can lead to faster training times. On the other hand, a small batch size can lead to a more stable training process, as the model is updated more frequently. There is a trade-off between big and small batch sizes, as a big batch size can lead to a faster convergence, but can also result in a suboptimal solution. On the other hand, a small batch size can lead to a slower convergence, but can result in a more accurate solution. In the field of deep learning, it has been shown that using a small batch size can result in better generalization performance. However, this is not always the case, as the optimal batch size depends on the specific problem and model being used. In conclusion, the choice of batch size is highly dependent on the specific problem and the model being used, and finding the optimal batch size requires experimentation and analysis [44].

• **Epochs**: In machine learning, an epoch is a single iteration through the entire training dataset. During each epoch, the model updates its parameters based on the gradient of the loss function calculated on the training data. Having a small number of epochs during training means that the model has seen the training data fewer times and thus has not had enough time to fully learn the relationships between the input features and output target. As a result, the model may not fit the training data well and is more likely to underfit the data. On the other hand, having a large number of epochs can lead to overfitting, where the model has memorized the training data and learned to fit it too well. As a result, the model may perform well on the training data but poorly on new, unseen data because it has learned to fit the noise and random fluctuations in the training data instead of the underlying patterns. Therefore, finding the right number of epochs for training is a trade-off between underfitting and overfitting, and the optimal number of epochs depends on the specific problem and dataset [37, 23].

• **Number of Neurons**: The number of neurons in a neural network is an important

hyperparameter that determines the complexity and capacity of the model. Having a small number of neurons in a neural network model can result in underfitting, as the model may not have enough capacity to learn the complex relationships between the input and output variables. This can lead to poor accuracy on the training and test data. On the other hand, having a large number of neurons can lead to overfitting, where the model memorizes the training data but performs poorly on new, unseen data. It is important to strike a balance between having too few and too many neurons in a neural network model [36].

• **Number of Layers**: Having a small number of layers in a neural network model can result in underfitting the data, as the model may not have enough capacity to capture the complexity of the problem. On the other hand, having a large number of layers can lead to overfitting, where the model becomes too specialized to the training data and doesn't generalize well to new data [36].

• **Activation Functions**: The best choice of activation function depends on the problem at hand and the type of model being used. Sigmoid and tanh are popular activation functions that work well for binary classification problems, but they can lead to vanishing gradients and slow training times for deep networks. ReLU (Rectified Linear Unit) is a commonly used activation function that has shown to work well for many deep learning problems. It has the advantage of having a non-saturating and linear response for positive inputs, which speeds up training and reduces the risk of vanishing gradients. Leaky ReLU is a variant of ReLU that helps address the issue of the "dying ReLU" problem, where the activation function becomes zero for negative inputs and the gradient vanishes. By allowing a small negative slope for negative inputs, leaky ReLU helps improve the performance of the model. In general, ReLU and its variants are a good choice for many problems, but it's important to experiment with different activation functions and choose the one that works best for the problem at hand [36] [35].

• **Optimizer**: When selecting an optimizer for training a machine learning model, it is important to consider both the strengths and weaknesses of each optimizer. The choice of optimizer can have a significant impact on the performance of the model.

Strengths:

SGD (Stochastic Gradient Descent) is a simple and widely used optimizer. It is computationally efficient and can handle large datasets. Adagrad and Adadelta adapt the learning rate for each weight based on the historical gradient information, allowing for more efficient optimization. RMSProp is similar to Adadelta but also incorporates moving averages, leading to a more stable convergence. Adam (Adaptive Moment Estimation) is a combination of RMSProp and momentum and has been shown to perform well in many applications. Nadam helps the optimization process move more efficiently towards the minimum, resulting in faster convergence. Nadam has the adaptive learning rate property of the Adam algorithm, which means it can adjust the learning rate based on the historical gradient information, helping to avoid oscillations and improve convergence.

Weaknesses:

SGD can have slow convergence and may get stuck in local minima. Adagrad and Adadelta may have a diminishing learning rate, leading to slow convergence. RMSProp and Adam may require careful tuning of hyperparameters such as learning rate and decay rate. Some optimizers such as Adam can be sensitive to the choice of learning rate and may require tuning. Nadam can be computationally expensive and can slow down the optimization process. The effectiveness of Nadam may depend on the choice of the hyperparameters, such as the learning rate and the decay rates, which can be difficult to tune.

It is recommended to experiment with different optimizers and evaluate their performance

on the specific task and dataset to determine the best optimizer [36] [27].

5.1 Optimizing Model Performance through Grid Search of Batch Size and Epoch Hyperparameters

In the field of machine learning, the performance of a model is highly dependent on the hyperparameters used during training. Grid search is a technique used to systematically search for the optimal combination of hyperparameters in a given range to achieve the best performance of a model. We will describe the process of conducting a grid search for the best batch size and epoch in the context of training a machine learning model.

Table 5.1: Table of trials during the grid search.

Trial	Batch Size	Epochs	Accuracy
1	500	200	96.57
2	500	400	96.76
3	500	800	96.98
4	1000	200	96.62
5	1000	400	96.58
6	1000	800	96.63
7	1500	200	96.42
8	1500	400	96.41
9	1500	800	96.33
10	2000	200	96.31
11	2000	400	96.46
12	2000	800	96.33

Table 5.1 table shows the results of 12 trials, where each trial has different batch size and a number of epochs. The trial number is listed in the first column, batch size in the second, number of epochs in the third, and accuracy in the fourth. The data in the table provides insight into the effect of changing batch size and a number of epochs on the accuracy of a machine learning model.The best accuracy is achieved when the batch size is 500 and the epoch is 800.

Figure 5.1 shows the accuracy in terms of the combination of Batch Size and Number of Epochs used for training the model. The highest accuracy is achieved for a batch size of 500 using 800 epochs. It's worth noting that other combinations may still produce acceptable results, but for optimal performance, the 500-800 combination seems to be the best choice for those combinations that were tried. Additionally, further optimization may yield even better results, so it's important to consider the trade-off between computational resources and accuracy when determining the optimal combination of batch size and a number of epochs.

5.2 Selective Search for Optimal Number of Neurons

Random search is an alternative approach to hyperparameter optimization in machine learning models. Instead of exhaustively searching over a predefined grid of hyperparameters, as

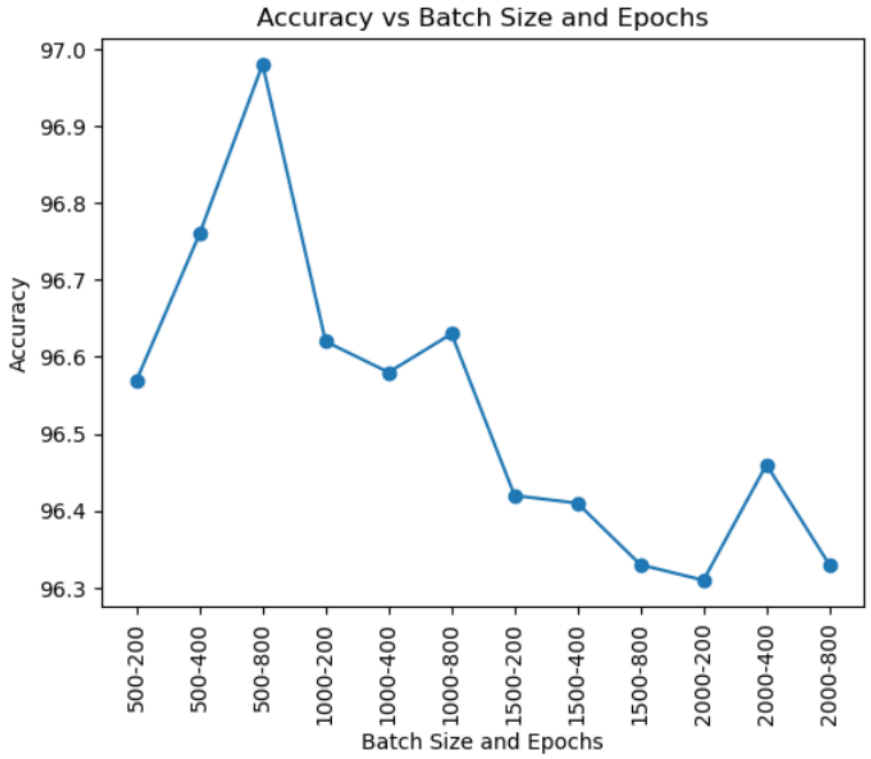

Figure 5.1: Accuracy in terms of the combination of Batch Size and Number of Epochs used for training the model. The highest accuracy is achieved for a batch size of 500 using 800 epochs.

in grid search, random search randomly samples hyperparameters from a probability distribution for a specified number of iterations. The idea behind random search is to explore the hyperparameter space more efficiently than grid search by focusing on high-performing regions and avoiding low-performing regions. Random search has been shown to perform similarly or even better than grid search in terms of finding the optimal hyperparameters in various applications [18] [47].

Here, instead of a random search, we will do a selective search by varying specific hyperparameters, fixing the batch size and epochs to the best found in the grid search for the reference model, i.e. 500 and 800, respectively and starting from the values of the reference model for the remaining hyperparameters.

Table 5.2 shows the accuracy of the machine-learning model when varying the number of neurons in the model. The highest accuracy was achieved when each hidden layer had 350 neurons. A corresponding plot is shown in Fig. 5.2.

5.3 Comparison of Optimization Algorithms for Improved Model Accuracy

In this subsection, we focus on finding the best optimizer for our model. Optimization is a crucial part of deep learning, as it helps the model to converge to a minimum of the loss function and achieve high accuracy. An optimizer helps to update the weights of the model in a direction that minimizes the loss function. Therefore, finding the right optimizer can

Table 5.2: Accuracy for different Number of Neurons per hidden layer. The highest accuracy is achieved for 350 neurons.

Number of neurons	**Accuracy**
50	95.11
100	95.78
150	95.98
200	95.88
250	96.43
300	96.47
350	96.70
400	96.60
450	96.53
500	96.66

greatly affect the performance of our model. In this subsection, we will perform experiments to compare the performance of different optimizers and determine the optimizer that yields the best results for our specific task. Other hyperparameters are fixed at 350 neurons per hidden layer, a batch size of 500 and 800 epochs.

Table 5.3: Accuracy achieved for difference choices of the Optimizer.

Optimizer	**Accuracy**
rmsprop	84.90
Nadam	96.61
Adam	96.49
Adamax	96.00
Adagrad	88.69
Adadelta	86.70

Table 5.3 shows the accuracy of a machine learning model when trained using different optimization algorithms. It follows that the Nadam and Adam optimizers have the highest accuracy with values of 96.61 and 96.49 respectively. These optimizers are variants of the popular Adam optimizer, which is known for its fast convergence and good performance. On the other hand, the rmsprop optimizer has a relatively lower accuracy with a value of 84.90. The Adagrad and Adadelta optimizers have intermediate accuracy values of 88.69 and 86.70 respectively. It's important to note that the accuracy values shown in this table are specific to the data and architecture used for the model. The optimal optimization algorithm for a given problem may depend on the specifics of the data and the architecture. A corresponding plot is shown in Figure 5.3.

5.4 Evaluating the Effect of Number of Layers on Model Performance

The purpose of this subsection is to identify the optimal number of layers in this neural network. To achieve this goal, various neural network architectures were tested and their performance was evaluated for accuracy. The experiments were carried out with a range of

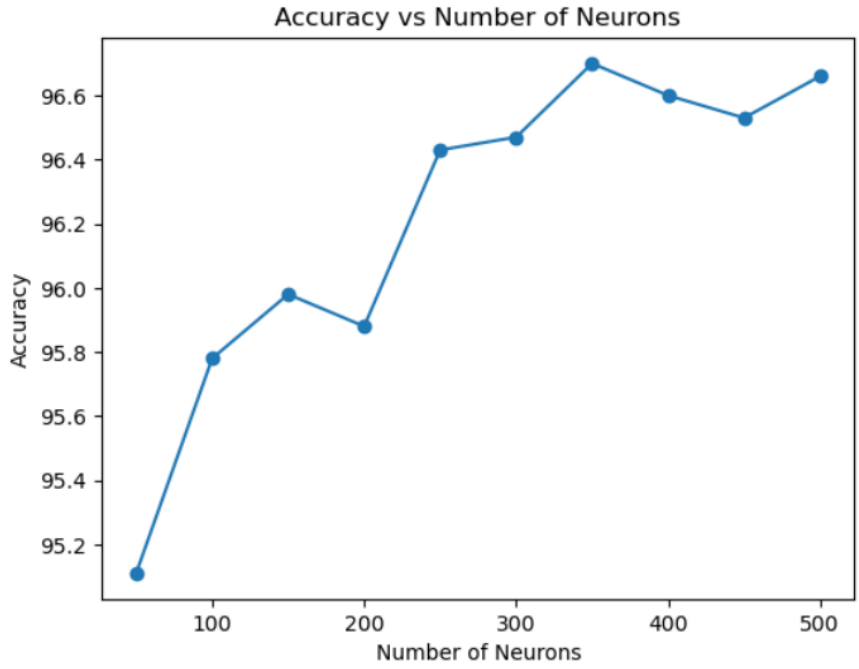

Figure 5.2: Accuracy as a function of the Number of Neurons per hidden layer. The highest accuracy is achieved for 350 neurons.

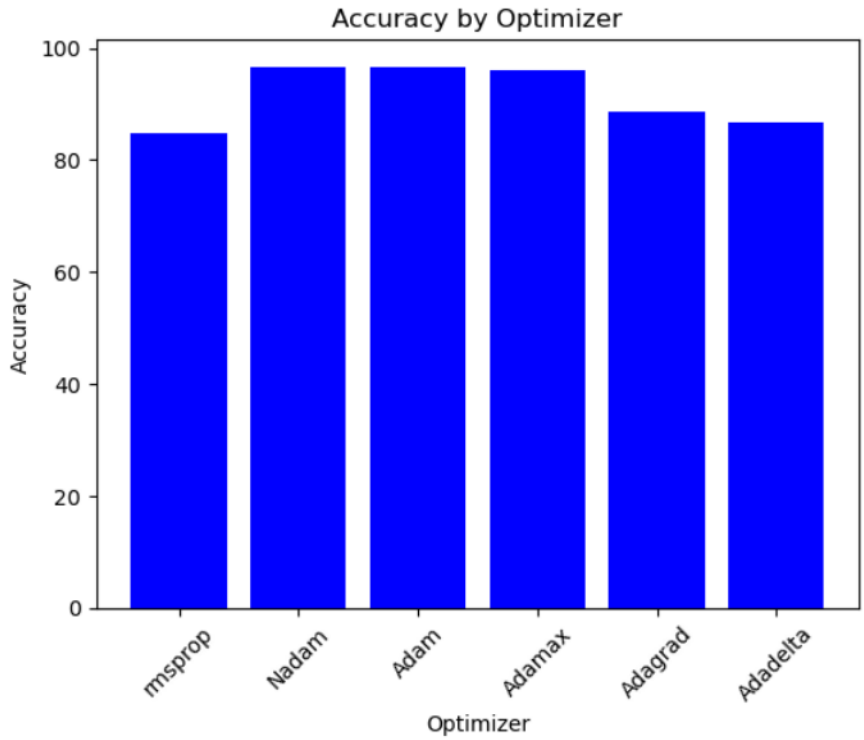

Figure 5.3: Bar graph showing the achieved accuracy for different choices of the optimizer (see text for details).

layer sizes, from 6 to 14, to determine the best number of layers for the network. The results of these experiments are presented in this subsection, and the optimal number of layers is determined based on the highest accuracy achieved.

Table 5.4: Accuracy as a function of the Number of Hidden Layers.

Number of hidden layers	**Accuracy**
6	96.61
8	96.66
10	96.96
12	91.98
14	91.98
16	84.03

Table 5.4 shows the achieved accuracy as function of the number of hidden layers, ranging from 6 to 16. The highest accuracy is achieved with 10 hidden layers with a value of 96.96%. For a larger number of hidden layers, we observe that the accuracy degrages significantly. The results suggest that there is an optimal number of hidden layers that can be used to achieve the best performance in this model.

5.5 Impact of Number of Data Points on Accuracy

In machine learning, having a larger dataset normally leads to better accuracy due to the model having more information to learn from. To investigate the impact of the number of data data points on accuracy, we experimented with four different datasets: 125k, 250k, 500k, and 1m.

Table 5.5: Accuracy for different number of data points.

Number of Data Points	**Accuracy**
125k	89.95
250k	93.55
500k	95.54
1M	96.96

In Table 5.5 we see that after training a model on each dataset, we found that the accuracy increased as the number of data points increased. The 125k dataset achieved an accuracy of 89.95%, while the 250k dataset achieved an accuracy of 93.55%. The 500k dataset achieved an accuracy of 95.54%, and the 1M dataset achieved the highest accuracy of all with 96.96%. A corresponding plot is shown in Figure 5.4, showing a nonlinear relation between the number of data points and the accuracy. It suggests that increasing the amount of training data can be an effective method for improving the accuracy of machine learning models. However, it's worth noting that having a larger dataset will also increase the time and computational resources required for training, so the trade-off between accuracy and resource consumption should be carefully considered when deciding on the size of the dataset.

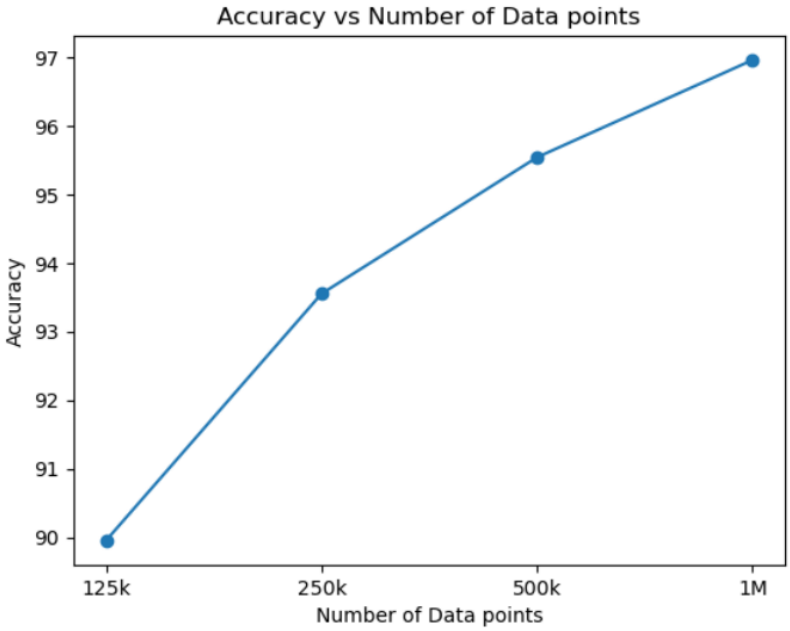

Figure 5.4: The Accuracy has a nonlinear relationship to the number of data points.

5.6 Extrapolation of Required Sample Size for Desired Accuracy

In order to achieve accurate results in machine learning models, it is important to have sufficient data. However, collecting, storing, and processing large amounts of data can be computationally expensive and time-consuming. Therefore, it is necessary to determine the minimum amount of data required to achieve a certain level of accuracy in a given machine learning task. Here, we will extrapolate from the results presented in the previous sections, in order to estimate the required number of data points to achieve an accuracy of 99%.

Figure 5.5 reveals a linear relationship between $\log(MAPE)$ and $\log(accuracy)$, allowing us to perform an extrapolation (via the least-squares method) to smaller desired values of MAPE than those achieved so far. The extrapolation shows that a dataset size of approximately 7 million data points will be required to reduce MAPE to 1% (equivaltently, to increase the accuracy to 99%).

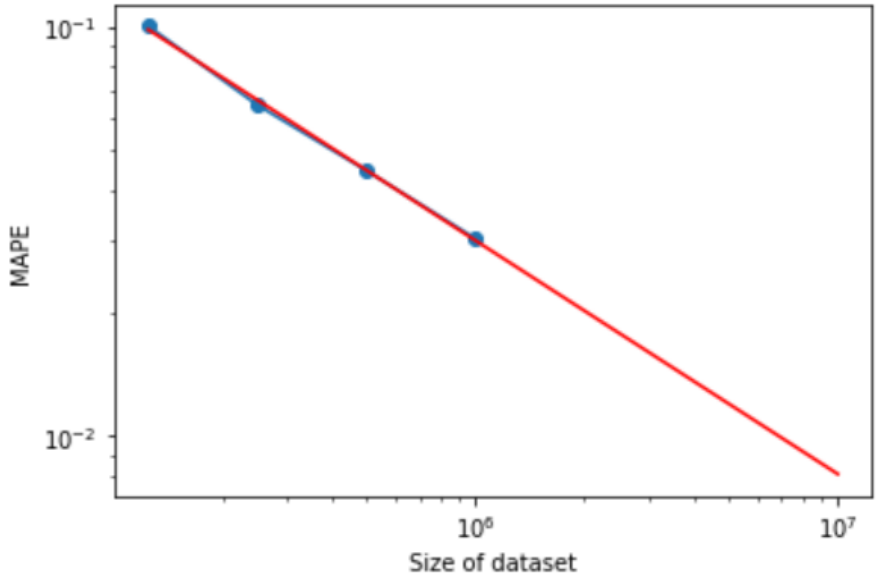

Figure 5.5: Log-log plot of MAPE vs. the number of points (size) of the dataset. A line is formed and its extrapolation shows that about 7 million data points are required to achieve a MAPE of 1% (equivalently, an accuracy of 99%).

Chapter 6

Discussion

Since 2015 we can now observe the dark side of the universe using interferometric gravitational wave detectors and nearly 90 binary black hole systems have already been detected. A large number of templates (of order hundreds of thousand) are needed to measure the source parameters of a detected system with perform matched filtering. In such parameter estimation calculations it is also useful to know the optimal signal-to-noise ratio of an individual model waveform. In this thesis, we trained an artificial neural network on a random sample of one million theoretical waveforms of binary black hole systems with random spins to predict the signal-to-noise ratio. We performed optimization of different hyperparameters with a grid search and also selectively searched for the accuracy of other parameter choices.

Our best model achieved an accuracy of 97% in predicting the signal-to-noise ratio, having the advantage of evaluating the results orders of magnitude faster (on a GPU) than the original calculation. By using different sizes of the dataset, we found that the logarithm of the accuracy is linearly related to the logarithm of the number of points in the dataset. This allowed us to predict that a dataset size of about 7 million data points will be required to achieve an accuracy of 99% in predicting the SNR with the neural network that we constructed.

Apart from increasing the datasize, one could also try other architectures that could potentially lead to higher accuracy. For example, ResNets allow for a larger number of layers in a deep neural network and it would be worth investigating whether they can lead to an increase in accuracy.

As part of this thesis, the Python code available at the following link:

GitHub repository

Bibliography

[1] J. Aasi et al. Advanced LIGO. *Classical and Quantum Gravity*, 32:074001, 2015.

[2] J Abadie, BP Abbott, R Abbott, TD Abbott, M Abernathy, T Accadia, F Acernese, C Adams, R Adhikari, C Affeldt, et al. All-sky search for gravitational-wave bursts in the second joint ligo-virgo run. *Physical Review D*, 85(12):122007, 2012.

[3] B. P. Abbott et al. The gravitational wave transient gw150914: Implications for the progenitor black hole mass and spin. *Physical Review X*, 6, 2016.

[4] B. P. Abbott et al. Gw170817: Observation of gravitational waves from a binary neutron star inspiral. *Physical Review Letters*, 119(16):161101, 2017.

[5] B. P. Abbott et al. Gravitational waves from compact binary coalescences: A review. *Living Reviews in Relativity*, 21, 2018.

[6] Benjamin P Abbott, Richard Abbott, TD Abbott, MR Abernathy, Fausto Acernese, Kendall Ackley, Carl Adams, Thomas Adams, Paolo Addesso, RX Adhikari, et al. Observation of gravitational waves from a binary black hole merger. *Physical review letters*, 116(6):061102, 2016.

[7] et al Abbott. Gwtc-1: A gravitational-wave transient catalog of compact binary mergers observed by ligo and virgo during the first and second observing runs. *Phys. Rev. X*, 9:031040, Sep 2019.

[8] R. Abbott et al. GWTC-2: Compact Binary Coalescences Observed by LIGO and Virgo During the First Half of the Third Observing Run. *Phys. Rev. X*, 11:021053, Jun 2021.

[9] R. Abbott et al. GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo During the Second Part of the Third Observing Run. *arXiv e-prints*, page arXiv:2111.03606, November 2021.

[10] F Acernese, F Barone, R De Rosa, A Eleuteri, L Milano, and R Tagliaferri. A hierarchical bayesian framework for nonlinearities identification in gravitational wave detector outputs. *Classical and Quantum Gravity*, 22(18):S1223, 2005.

[11] F Acernese et al. Advanced Virgo: a second-generation interferometric gravitational wave detector. *Classical and Quantum Gravity*, 32(2):024001, 2014.

[12] Fausto Acernese, P Amico, N Arnaud, C Arnault, D Babusci, G Ballardin, F Barone, M Barsuglia, F Bellachia, JL Beney, et al. The present status of the virgo central interferometer. *Classical and Quantum Gravity*, 19(7):1421, 2002.

[13] MEJBAH AHAMMAD. Building a regression multi-layer perceptron (mlp) — kaggle.

[14] T Akutsu et al. Overview of KAGRA: Detector design and construction history. *Progress of Theoretical and Experimental Physics*, 2021(5), 08 2020. 05A101.

[15] ZA Allen, P Astone, L Baggio, D Busby, M Bassan, DG Blair, Michele Bonaldi, P Bonifazi, P Carelli, M Cerdonio, et al. First search for gravitational wave bursts with a network of detectors. *Physical review letters*, 85(24):5046, 2000.

[16] Pau Amaro-Seoane, Heather Audley, Stanislav Babak, John Baker, Enrico Barausse, Peter Bender, Emanuele Berti, Pierre Binetruy, Michael Born, Daniele Bortoluzzi, Jordan Camp, Chiara Caprini, Vitor Cardoso, Monica Colpi, John Conklin, Neil Cornish, Curt Cutler, Karsten Danzmann, Rita Dolesi, Luigi Ferraioli, Valerio Ferroni, Ewan Fitzsimons, Jonathan Gair, Lluis Gesa Bote, Domenico Giardini, Ferran Gibert, Catia Grimani, Hubert Halloin, Gerhard Heinzel, Thomas Hertog, Martin Hewitson, Kelly Holley-Bockelmann, Daniel Hollington, Mauro Hueller, Henri Inchauspe, Philippe Jetzer, Nikos Karnesis, Christian Killow, Antoine Klein, Bill Klipstein, Natalia Korsakova, Shane L Larson, Jeffrey Livas, Ivan Lloro, Nary Man, Davor Mance, Joseph Martino, Ignacio Mateos, Kirk McKenzie, Sean T McWilliams, Cole Miller, Guido Mueller, Germano Nardini, Gijs Nelemans, Miquel Nofrarias, Antoine Petiteau, Paolo Pivato, Eric Plagnol, Ed Porter, Jens Reiche, David Robertson, Norna Robertson, Elena Rossi, Giuliana Russano, Bernard Schutz, Alberto Sesana, David Shoemaker, Jacob Slutsky, Carlos F. Sopuerta, Tim Sumner, Nicola Tamanini, Ira Thorpe, Michael Troebs, Michele Vallisneri, Alberto Vecchio, Daniele Vetrugno, Stefano Vitale, Marta Volonteri, Gudrun Wanner, Harry Ward, Peter Wass, William Weber, John Ziemer, and Peter Zweifel. Laser Interferometer Space Antenna. *arXiv e-prints*, page arXiv:1702.00786, February 2017.

[17] P Astone, D Babusci, M Bassan, P Bonifazi, E Coccia, S D'Antonio, V Fafone, G Giordano, A Marini, Y Minenkov, et al. The next science run of the gravitational wave detector nautilus. *Classical and Quantum Gravity*, 19(7):1911, 2002.

[18] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[19] Léon Bottou. From machine learning to machine reasoning: An essay. *Machine learning*, 94:133–149, 2014.

[20] Carlton M Caves, Kip S Thorne, Ronald WP Drever, Vernon D Sandberg, and Mark Zimmermann. On the measurement of a weak classical force coupled to a quantum-mechanical oscillator. i. issues of principle. *Reviews of Modern Physics*, 52(2):341, 1980.

[21] David F Chernoff and Lee Samuel Finn. Gravitational radiation, inspiraling binaries, and cosmology. *arXiv preprint gr-qc/9304020*, 1993.

[22] F Chollet and JJ Allaire. Deep learning with r. greenwich, ct, 2018.

[23] Francois Chollet. *Deep Learning with Python*. Manning Publications, 2015.

[24] LIGO Scientific Collaboration. Mass ratio and the character of gravitational wave signals. *The Astrophysical Journal*, 826(2):L13, 2016.

[25] Curt Cutler and Eanna E Flanagan. Gravitational waves from merging compact binaries: How accurately can one extract the binary's parameters from the inspiral waveform? *Physical Review D*, 49(6):2658, 1994.

[26] Karsten Danzmann, Harald Lück, Albrecht Rüdiger, Roland Schilling, M Schrempel, Walter Winkler, Jim Hough, GP Newton, NA Robertson, DI Robertson, et al. Geo 600. a 600 m laser interferometric gravitational wave antenna. In *Edoardo Amaldi Conference on Gravitational Wave Experiments*, pages 100–111, 1995.

[27] Timothy Dozat. Incorporating nesterov momentum into adam. *International Conference on Learning Representations*, 2016.

[28] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[29] Lee S Finn. Detection, measurement, and gravitational radiation. *Physical Review D*, 46(12):5236, 1992.

[30] Lee Samuel Finn and David F Chernoff. Observing binary inspiral in gravitational radiation: One interferometer. *Physical Review D*, 47(6):2198, 1993.

[31] Frederic B. Fitch. Warren s. mcculloch and walter pitts. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133. *Journal of Symbolic Logic*, 9(2):49–50, 1944.

[32] Chris L Fryer and Vassiliki Kalogera. Theoretical black hole mass distributions. *The Astrophysical Journal*, 554(1):548, 2001.

[33] Chris L Fryer and Kimberly CB New. Gravitational waves from gravitational collapse. *Living Reviews in Relativity*, 14(1):1–93, 2011.

[34] A Giazotto et al. The virgo experiment: status of the art. In *First Edoardo Amaldi Conference on Gravitational Wave Experiments*, volume 1, page 86. World Scientific, Singapore, 1995.

[35] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. *AISTATS*, pages 315–323, 2010.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. *MIT Press*, 2016.

[38] L Gottardi, A De Waard, O Usenko, G Frossati, M Podt, Jakob Flokstra, M Bassan, V Fafone, Y Minenkov, and A Rocchi. Sensitivity of the spherical gravitational wave detector minigrail operating at 5 k. *Physical Review D*, 76(10):102005, 2007.

[39] R. M. Green. *Spherical Astronomy*. Cambridge University Press, 1987.

[40] L Grippo and M Sciandrone. Metodi di ottimizzazione per le reti neurali. *Università di Roma La Sapienza e Consiglio nazionale delle ricerche, Roma*, 2006.

[41] Carl W Helstrom. *Statistical theory of signal detection: international series of monographs in electronics and instrumentation*, volume 9. Elsevier, 2013.

[42] Robert C Hilborn. Gw170814: Gravitational wave polarization analysis. *arXiv preprint arXiv:1802.01193*, 2018.

[43] Piotr Jaranowski and Andrzej Królak. Gravitational-wave data analysis. formalism and sample applications: The gaussian case. *Living Reviews in Relativity*, 8:1–38, 3 2005.

[44] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[46] Laboratori Nationali Legnaro. Auriga bar detector. *project homepage. URL (cited on 08 November 2007): http://www. auriga. lnl. infn. it*, 1(4.1).

[47] Luyu Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Alex Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 97–106, 2017.

[48] Harald Lück, GEO600 Team, et al. The geo600 project. *Classical and quantum gravity*, 14(6):1471, 1997.

[49] Michele Maggiore. Gravitational Waves. Vol. 1: Theory and Experiments. *Oxford University Press*, October 2018.

[50] NASA. Binary black hole gravitational waves, N/A. Image Credit: NASA.

[51] David Nicholson, CA Dickson, W John Watkins, Bernard F Schutz, J Shuttleworth, GS Jones, DI Robertson, NL Mackenzie, Kenneth A Strain, BJ Meers, et al. Results of the first coincident observations by two laser-interferometric gravitational wave detectors. *Physics Letters A*, 218(3-6):175–180, 1996.

[52] Eleftherios Papantonopoulos. *The Physics of the Early Universe*, volume 653. Springer Science & Business Media, 2005.

[53] Allan Pinkus. Tdi-subspaces ofc (rd) and some density problems from neural networks. *journal of approximation theory*, 85(3):269–287, 1996.

[54] MV Plissi, KA Strain, CI Torrie, NA Robertson, S Killbourn, S Rowan, SM Twyford, H Ward, KD Skeldon, and J Hough. Aspects of the suspension system for geo 600. *Review of Scientific Instruments*, 69(8):3055–3061, 1998.

[55] Giovanni Andrea Prodi, Marco Drago, and Serena Vinciguerra. Gravitational wave searches for transient signals: classication of candidates consistent with binary black hole mergers rapporteur, 2012.

[56] FJ Raab. The ligo project: progress and prospects. In *First Edoardo Amaldi Conference on Gravitational Wave Experiments*, pages 70–85. World Scientific, Singapore, 1995.

[57] Salvatore Rampone, Vincenzo Pierro, Luigi Troiano, and Innocenzo M Pinto. Neural network aided glitch-burst discrimination and glitch classification. *International Journal of Modern Physics C*, 24(11):1350084, 2013.

[58] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[59] F Rosenblatt. Spartan books. *Washington, DC*, 1962.

[60] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[61] Paul Sajda. Neural networks. *Encyclopedia of the Human Brain*, pages 373–383, 2002.

[62] Bangalore Suryanarayana Sathyaprakash and Bernard F Schutz. Physics, astrophysics and cosmology with gravitational waves. *Living reviews in relativity*, 12(1):1–141, 2009.

[63] Marlin Benedikt Schäfer. Machine learning applications in search algorithms for gravitational waves from compact binary mergers. *PhD Thesis, University of Jena*, 2023.

[64] Bernard F Schutz. Data processing, analysis, and storage for interferometric antennas. In *The detection of gravitational waves*, pages 406–451. Cambridge University Press, 1991.

[65] Bernard F Schutz. *Gravitational wave data analysis*, volume 253. Springer Science & Business Media, 2012.

[66] K. S. Thorne. Disk accretion onto a black hole. 2. evolution of the hole. *The Astrophysical Journal*, 248(2):507–520, 1974.

[67] Kip S Thorne. Gravitational waves. *arXiv preprint gr-qc/9506086*, 1995.

[68] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[69] Anna L Watts, Badri Krishnan, Lars Bildsten, and Bernard F Schutz. Detecting gravitational wave emission from the known accreting neutron stars. *Monthly Notices of the Royal Astronomical Society*, 389(2):839–868, 2008.

[70] Karlijn Willems. Keras tutorial: Deep learning in python, 2017.

[71] Benno Willke, P Ajith, B Allen, P Aufmuth, C Aulbert, S Babak, Ramachandran Balasubramanian, BW Barr, S Berukoff, A Bunkowski, et al. The geo-hf project. *Classical and Quantum Gravity*, 23(8):S207, 2006.

[72] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.