MASARYKOVA UNIVERZITA
Přírodovědecká fakulta
Ústav teoretické fyziky a astrofyziky

Diplomová práce

Brno 2023 **Martin Chobola**

MASARYKOVA
UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
ÚSTAV TEORETICKÉ FYZIKY A ASTROFYZIKY

Machine learning enhanced search for transient events in light curves

Diplomová práce

Martin Chobola

Vedoucí práce: Dr. Martin Topinka, PhD. **Brno 2023**

Bibliografický záznam

Autor:	Martin Chobola Přírodovědecká fakulta, Masarykova univerzita Ústav teoretické fyziky a astrofyziky

Název práce:	Machine learning enhanced search for transient events in light curves
Studijní program:	Fyzika
Studijní obor:	Astrofyzika
Vedoucí práce:	Dr. Martin Topinka, PhD.
Akademický rok:	2022/23
Počet stran:	56 + 1
Klíčová slova:	gamma záblesk,strojové učení, CubeSats, detekce transientů, LSTM, konvoluční sítě

Bibliographic Entry

Author:	Martin Chobola Faculty of Science, Masaryk University Department of Theoretical physics and Astrophysics

Title of Thesis:	Machine learning enhanced search for transient events in light curves
Degree Programme:	Physics
Field of Study:	Astrophysics
Supervisor:	Dr. Martin Topinka, PhD.
Academic Year:	2022/23
Number of Pages:	56 + 1
Keywords:	gamma-ray burst,machine learning,CubeSats, transiant detection, LSTM, convolutional network

Abstrakt

Detekce gama záblesků (GRB) v družicových datech s vysokým pozadím zůstává v astrofyzice náročným problémem. V této práci se zabývám využitím algoritmů strojového učení k detekci GRB v pozorováních shromážděných družicí GRBAlpha. Porovnávám výkonnost několika nejmodernějších algoritmů strojového učení: odšumovacího autoenkodéru, predikce pozadí založené na LSTM a binárního klasifikátoru založeného na kombinaci LSTM a konvolučních vrstev. Hledám optimální prahovou hodnotu, která vyvažuje pravdivé a falešně pozitivní výsledky.

Když byl nejlepší model aplikován na data za dva roky pozorování, dosáhl 100% přesnosti při vyhledávání známých GRB s poměrem signálu k šumu větším než 3.

Moje výsledky ukazují, že algoritmy strojového učení mohou výrazně zlepšit detekci GRB v zašuměných družicových datech a překonat standardní přístup detekce odlehlých hodnot. Tuto metodu lze implementovat do družic pro systémy detekce transientů v reálném čase a offline vyhledávání v archivních datech.

Abstract

The detection of Gamma-Ray Bursts (GRBs) in the noisy background of satellite data remains a challenging problem in astrophysics. In this thesis, I explore the use of machine learning algorithms to detect GRBs in observations collected by the GRBAlpha satellite. I compare the performance of several state-of-art machine learning algorithms: denoising autoencoder, LSTM-based background prediction, and a binary classifier based on the combination of LSTM and convolutional layers. I search for optimal threshold, balancing true and false positives.

When the best model was applied to two years' worth of data, it reached 100% accuracy in finding known GRBs with signal-to-noise ratio greater than 3.

My results demonstrate that machine learning algorithms can significantly improve the detection of GRBs in noisy satellite data, beating the standard outlier z-score detection approach. This method can be implemented into satellites for real-time transient detection triggering systems and offline searches on archival data.

MUNI
SCI

MASARYKOVA UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
KOTLÁŘSKÁ 2, 611 37 BRNO
IČ: 00216224
DIČ: CZ00216224

ZADÁNÍ
DIPLOMOVÉ PRÁCE

Akademický rok: 2022/2023

Ústav:	Ústav teoretické fyziky a astrofyziky
Student:	Bc. Martin Chobola
Program:	Fyzika
Specializace:	Astrofyzika

Ředitel *ústavu* PřF MU Vám ve smyslu Studijního a zkušebního řádu MU určuje diplomovou práci s názvem:

Název práce:	Machine learning enhanced search for transient events in light curves
Název práce anglicky:	Machine learning enhanced search for transient events in light curves
Jazyk závěrečné práce:	angličtina

Oficiální zadání:
The fast growing observation rate of the restless Universe, including the recent boom of small-size satellites at the low Earth orbit and sky monitors with large field of view, bring new challenges in the search for transients in the data, especially in the moderate to high background environment. The goal of the thesis would be to study and investigate new approaches in flare detection using machine learning techniques. The new methods will be tested on simulated data and on the real examples of the photometric data from existing space missions. If calibrated well the method has capabilities to become an important tool in transient searches in upcoming fleets of space missions. Cosmic gamma-ray bursts, alone or as a part of the search for gravitational wave counterparts is just one case of many possible fields of usage.

Vedoucí práce:	Dr. Martin Topinka, PhD.
Konzultant:	RNDr. Jakub Řípa, Ph.D. Mgr. Tomáš Plšek
Datum zadání práce:	18. 11. 2021
V Brně dne:	30. 1. 2023

Zadání bylo schváleno prostřednictvím IS MU.

Bc. Martin Chobola, 9. 1. 2022
Dr. Martin Topinka, PhD., 11. 1. 2022
Mgr. Dušan Hemzal, Ph.D., 25. 1. 2022

Poděkování

Díky moc všem!

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně s využitím informačních zdrojů, které jsou v práci citovány.

Brno 6. června 2023

. . . . . .
Martin Chobola

Contents

Chapter 1

Gamma-ray Bursts

Gamma-ray bursts (GRBs) are some of the most energetic and unpredictable events in the universe. These events occur at random times and positions in the sky (Figure 1.1) (Fishman et al., 1982) and can last from milliseconds to several minutes (Figure 1.2), with a rate of several per day per sky and energy $E_{\rm iso}$[1] up to 10^{54} erg, spectrum peaking for sGRBs in hard x-rays and long GRBs in gamma-rays. Since gamma rays do not penetrate the Earth's atmosphere, they are detected by satellites, which automatically trigger ground-based telescopes for follow-up observations at longer wavelengths (Gomboc, 2012).

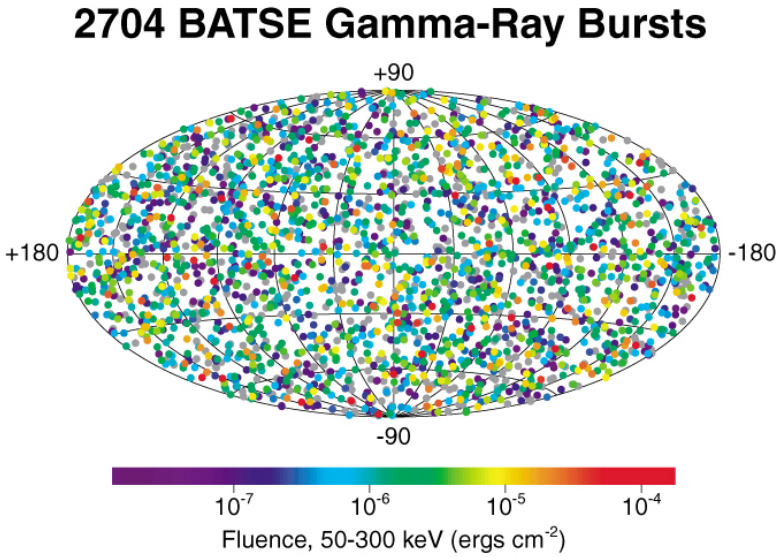


Figure 1.1: Positions on the sky of all gamma-ray bursts detected during the BATSE mission. The distribution is isotropic, with no concentration in any particular area[2].

GRBs can be classified based on their duration T_{90}[3] into two categories: short and long. Short GRBs (SGRB) typically last for less than two seconds but can be short as a few milliseconds, while long GRBs (LGRB) last for several minutes (Kouveliotou et al., 1993) or exceptionally hours (Martin-Carrillo et al., 2014). These durations are closely related to the progenitor of the GRB. The SGRBs are thought to arise from the merger of two compact objects, such as two neutron stars or a neutron star and a black hole (Eichler et al., 1989; Narayan et al., 1992; Berger, 2014),

[1] $E_{\rm iso}$ represents the energy emitted in all directions, assuming that the burst radiation is isotropically distributed and corrected by beaming effect (GRBs emit in highly collimated jet, Kumar & Zhang, 2015) and cosmological redshift.

[2] Image taken from https://heasarc.gsfc.nasa.gov/docs/objects/heapow/archive/transients/batse_bursts.html.

[3] T_{90} is a commonly used parameter to describe the duration of the prompt emission phase of a GRB, defined as the time interval between the times at which 5% and 95% of the total burst fluence are emitted making it the time duration during which the burst releases the majority of its emission in the given energy band. The determination of T_{90} is not exact, it depends on the instrument, background noise, energy band and burst intensity.

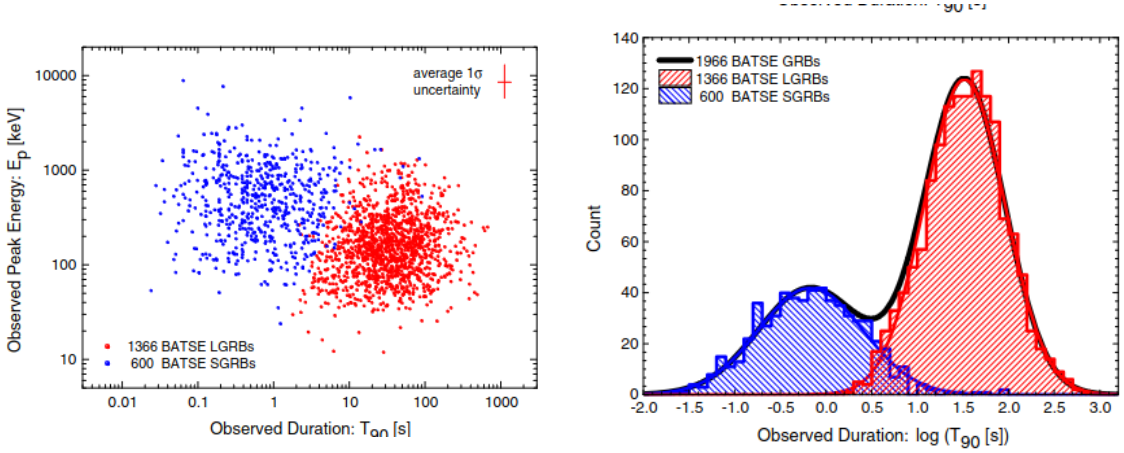


Figure 1.2: **Left:** distribution of the observed duration, **right:** distribution of the observed duration to spectral hardness (hard/soft) (Shahmoradi, 2013).

their millisecond variability further suggests that the regions from which the energy originated are small in size. For long GRBs, the progenitor is believed to be associated with the core collapse of a massive star (Woosley, 1993; MacFadyen & Woosley, 1999). Detection of gravitational waves from binary neutron star merger GW170817 by LIGO and Virgo observatories (Abbott & et al., 2017) and simultaneous detection of short GRB (GRB170817A) by many instruments including *Fermi-GBM* and *INTEGRAL* (Goldstein et al., 2017) has confirmed this mechanism for the SGRBs and started a new multi-messenger era.

This bimodality is also supported by the *hardness ratio*, which is defined as the ratio of counts in the high-energy band to the low-energy band. Generally, this ratio is computed as the ratio of the number of counts in the $50 - 300$ keV band to the number of counts in the $25 - 50$ keV band, but the ranges may differ depending on the instrument. As we can see, short-duration GRBs tend to have harder spectra than long-duration GRBs (Ghirlanda et al., 2004). Unfortunately, this is not as simple. There are four main setbacks: (i) the difference in brightness between the strongest and weakest bursts causes variations in measured duration (Norris et al., 2000), (ii) far-away bursts look longer due to the cosmological time dilation (Lamb & Reichart, 2000), (iii) the energy band observed affects the duration of short bursts which usually have only a few pulses (Norris et al., 1996; Kazanas et al., 1998), and (iv) some of the shortest bursts may not be detected because of instrument limitations (Lee & Petrosian, 1996).

As you can see in Figure 1.2, there is an overlap of duration and hardness. This overlap challenges the notion of a clear-cut division and suggests a more nuanced understanding of GRBs. It is now understood that certain long-duration GRBs can exhibit characteristics commonly associated with short-duration GRBs and vice versa, e.g. GRBs with $T_{90} >> 2\,\mathrm{s}$ may also originate from compact object merger. For several long GRBs (e.g., GRB 060605 and GRB 060614) deep optical observations excluded an accompanying supernova (Fynbo et al., 2006). Furthermore, GRBs with short peaked gamma-ray emission followed by a spectrally softer extended emission (EE-SGRBs) have been proposed to originate from mergers of compact objects (Norris, 2002; Norris & Bonnell, 2006; Gehrels et al., 2006).

As seen in Figure 1.1, GRBs are isotropically spatially distributed (Kouveliotou et al., 1993; Řípa & Shafieloo, 2019). The rate of GRBs is $33 \pm 11\ \mathrm{Gpc}^{-3}\ \mathrm{yr}^{-1}$, which corresponds to roughly once a day, but may vary depending on the instrument and distance in interest. They are named based on the date they were detected, using the format YYMMDD (year, month, day), followed by a letter with the order they were detected during that day.

1.1 Physics behind GRBs

A GRB event can be divided into two phases (Figure 1.3): prompt emission and afterglow. In the generally accepted fireball model (Narayan et al., 1992; Piran, 2005), the initial burst of high-energy gamma-rays is called a **prompt emission**, and it is produced by the collision of relativistic shells of material ejected from the progenitor within the jet. This creates a shock wave that accelerates particles to emit gamma rays through synchrotron radiation and amplifies the magnetic field at the shock. The prompt emission is characterized by a complex light curve with multiple peaks and a non-thermal spectrum (Dai et al., 2017).

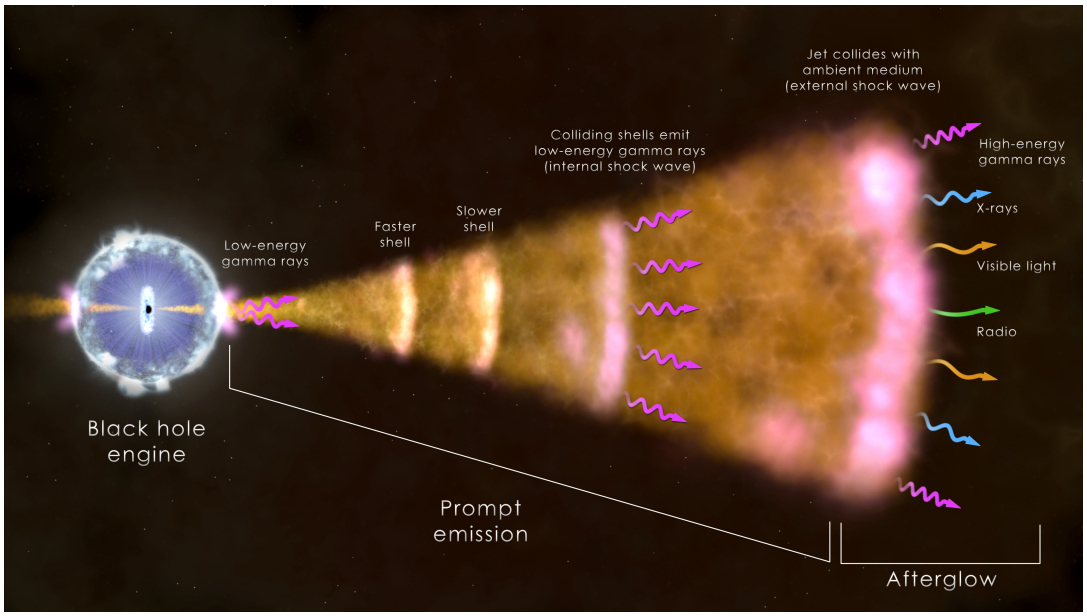


Figure 1.3: Radiation across the spectrum arises from hot ionized gas (plasma) in the vicinity ($\sim 10^9 - 10^{12}$ cm) of the progenitor, collisions among shells of fast-moving gas within the jet (internal shock waves), and from the leading edge of the jet as it sweeps up and interacts with its surroundings (external shock)[4].

Gamma-ray burst light curves exhibit an extensive range of complexity and diversity (Figure 1.4), with each burst having a distinct and unique profile. Observable emission can last anywhere from milliseconds to several minutes or even hours, and the shape of the light curve can vary greatly, with some bursts featuring a single peak while others have multiple sub-pulses (Figure 1.4). Additionally, individual peaks may have symmetrical shapes or display rapid brightening followed by slow fading, known as the Fast Rise-Exponential Decay profile (FRED). Some bursts are preceded by a "precursor" event, which is a weaker burst that is followed by a much more intense episode after a period of no emission lasting seconds to minutes. There are some events that have highly complex and chaotic light curve profiles with no clear discernible patterns.

[4] Image taken from https://www.nasa.gov/sites/default/files/thumbnails/image/grb_shell_final_0.jpg.

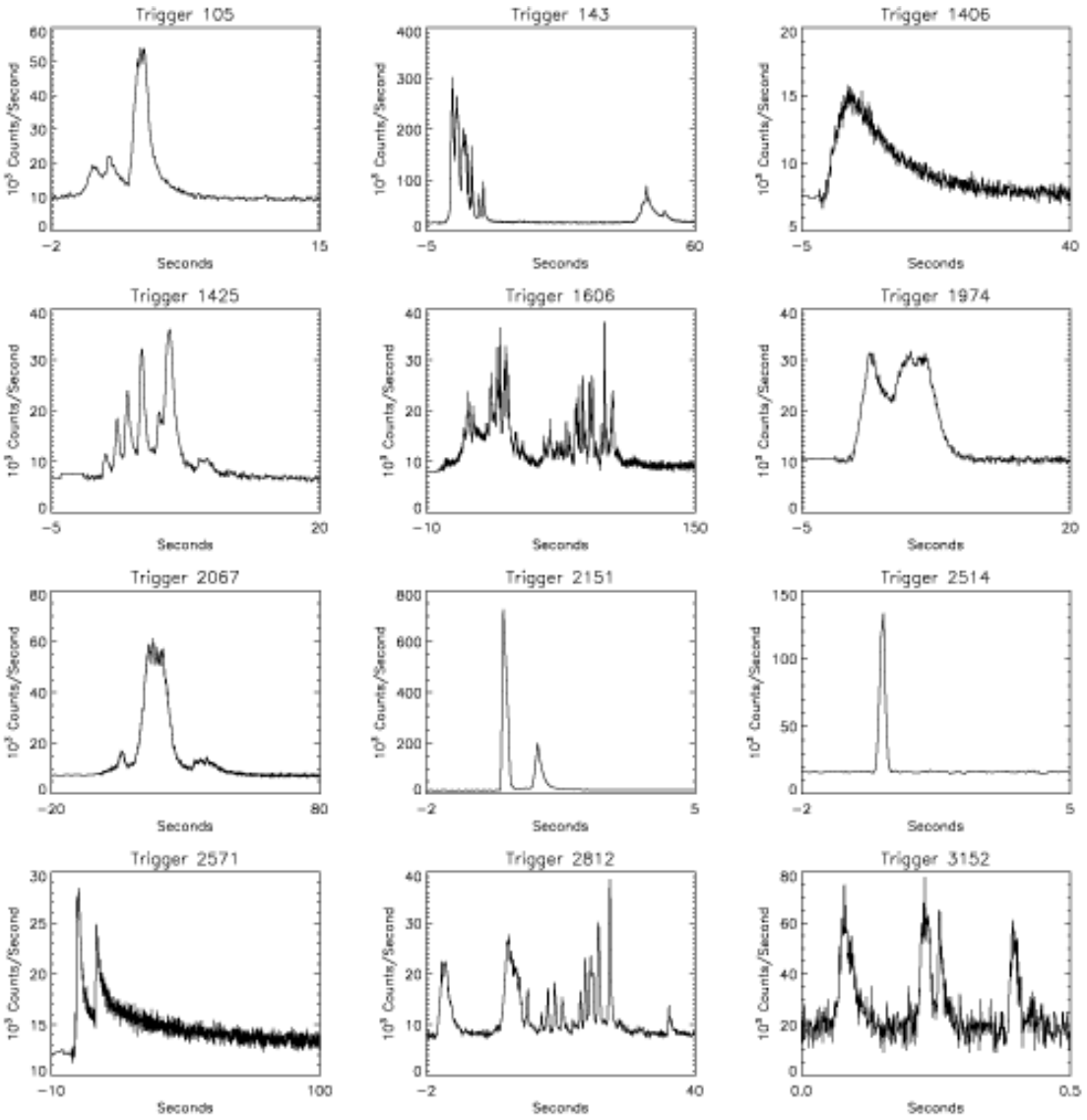


Figure 1.4: Clear representation of GRB profile diversity with data from the first BATSE catalog (Band et al., 1993)[5].

The energies and fluences associated with gamma-ray bursts (GRBs) surpass those of all other astrophysical phenomena by several orders of magnitude. GRB fluences typically span the range of $10^{-4} - 10^{-7}$: erg, cm^{-2}, corresponding to energy levels of $10^{51} - 10^{52}$: erg. These immense energy levels place GRBs at cosmological distances, ranging from hundreds of megaparsecs to units of gigaparsecs, as determined through redshift measurements from afterglows. To put this into perspective, the energy released in a GRB is comparable to that of a supernova explosion (approximately 10^{51} erg) or the total energy our Sun emits over its entire lifetime, but compressed into a mere few milliseconds.

[5] Image taken from CGRO BATSE https://heasarc.gsfc.nasa.gov/docs/objects/grbs/grb_profiles.html.

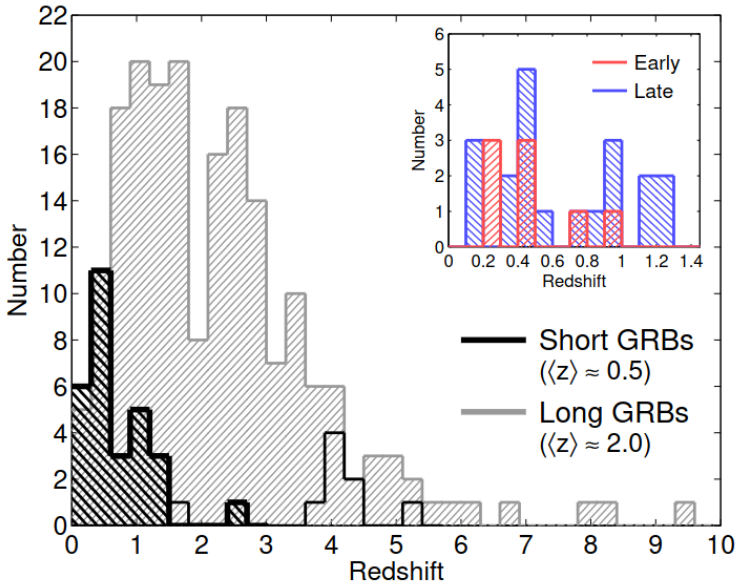


Figure 1.5: The redshift distribution of short GRBs (black) and long GRBs (grey). The inset shows the redshift distribution of short GRBs separated by host galaxy type, which exhibits no discernible difference between early-type (red) and late-type (blue) hosts (Berger, 2014).

Timely identification and localization of prompt emission are crucial for follow-up observation of an afterglow emission in multiple wavelengths if any (Greiner et al., 2010). The information about newly triggered GRBs is traditionally swiftly distributed to the GRB community by the online GRB Coordinates Network (GCN) to allow a rapid follow-up observation from ground instruments (Barthelmy, 2003).

Afterglows are the long-lasting emissions of radiation that follow the initial burst of gamma rays. After the initial burst, the afterglow can be observed in various wavelengths, including X-rays, visible light, and radio waves; the afterglow hunts are also reported through the GCN channel. The afterglow emission is caused by synchrotron radiation, which occurs when high-energy electrons are accelerated in a magnetic field. In the case of GRBs, the magnetic field is generated by the shock wave that is produced when the relativistic ejecta from the explosion interact with the surrounding medium. As the shock wave propagates through the medium, it accelerates particles to very high energies, producing synchrotron radiation. The characteristics of the afterglow depend on a number of factors, including the energy and duration of the initial burst, the density of the surrounding medium, the distance between the GRB and the observer, and also a viewing angle. By studying the afterglow emission, astronomers can learn about the properties of the GRB itself, as well as the properties of the surrounding medium (Mészáros & Rees, 1997; van Paradijs et al., 2000).

The extreme release of energy during these events can reveal insights into a wide range of fundamental physics processes and making them an important laboratory for studying the behavior of high-energy particles and fields in extreme conditions. Acceleration of particles to very high speeds, often approaching the speed of light, makes them a prime target for studying the properties of relativistic motion, such as time dilation, length contraction, and the Lorentz invariance violation. The observed time delays between different components of the GRB emission can provide information about the properties of the relativistic outflows that produce the bursts (Piran, 2005). Since GRBs originate at cosmological distances, the penetrating power of their emission and linkage to compact objects can serve as a proxy to measure the star formation rate. Their spectra can trace the intergalactic medium on the way from the source to us (Prochaska

et al., 2007). The detection of gravitational waves from the merger of neutron stars associated with a GRB in 2017 (GRB170817a; Abbott & et al., 2017) provided the first multi-messenger detection and allowed for tests of general relativity. The detailed study of the afterglow emission can help to constrain the properties of the explosion and the environment in which it occurs, while multi-messenger observations can provide complementary information about the physics of the event. Overall, the study of GRBs offers a unique and exciting opportunity to probe some of the most fundamental physical processes in the universe, providing new insights into the behavior of matter, energy, and space-time under extreme conditions.

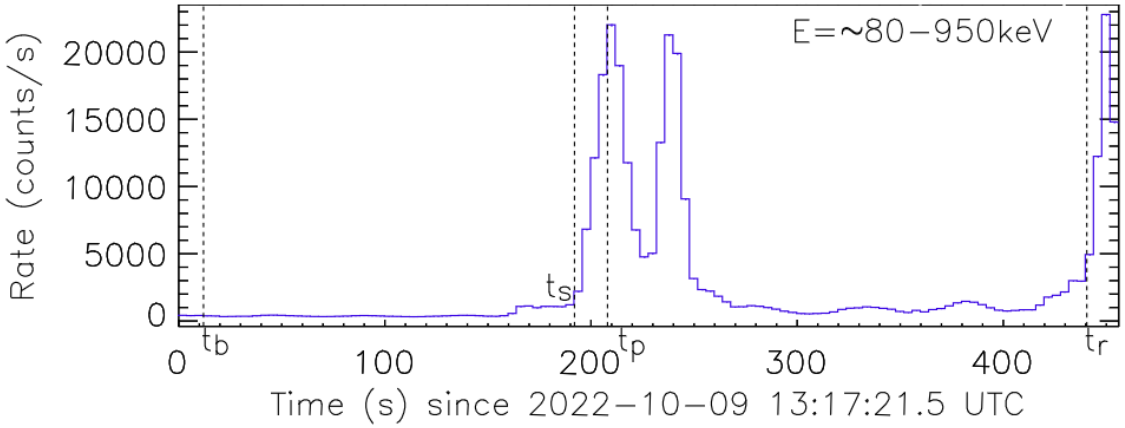


Figure 1.6: Light curve of the Brightest of All Time (BOAT) gamma-ray burst GRB221009A detected by GRBAlpha (Ripa et al., 2023).

1.2 Detection strategies

Gamma-ray detectors in space detect extreme bursts from astronomical objects such as supernovae, pulsars, and active galactic nuclei, represented as a sudden peak of counts on a detector. However, they can also detect background radiation that is present in the environment around the detector with a variety of sources, including the Earth's magnetic field, the Sun, cosmic rays, and even the detector itself, and needs to be accounted for. The background levels of gamma-ray detectors in space can vary depending on a number of factors, including the energy range of the gamma-rays being detected, the location of the detector in space, and the sensitivity of the detector itself (Hughes, 2004). To establish a baseline level of background, analysis of data before and after the peak is done, which is then subtracted, giving us information about the gamma-ray emission. Once the background noise is subtracted, analysis of the remaining gamma-ray signal is done to determine if it exhibits certain characteristics typical of GRBs. These characteristics include a FRED profile (Figure 1.4), a distinct spectral shape (the distribution of energies), and its duration (Figure 1.2).

In general, gamma-ray detectors that are located in Low-Earth Orbit (LEO) will experience higher background levels than those in higher orbits or in deep space due to a number of factors. For example, the polar regions of LEO are areas where charged particles from the Earth's radiation belts can become trapped, creating a high-radiation environment. Similarly, the South Atlantic Anomaly (SAA) is a region where the Earth's magnetic field is weakest, allowing charged particles from the radiation belts to penetrate lower altitudes (Koskinen & Kilpua, 2021; Baker et al., 2017).

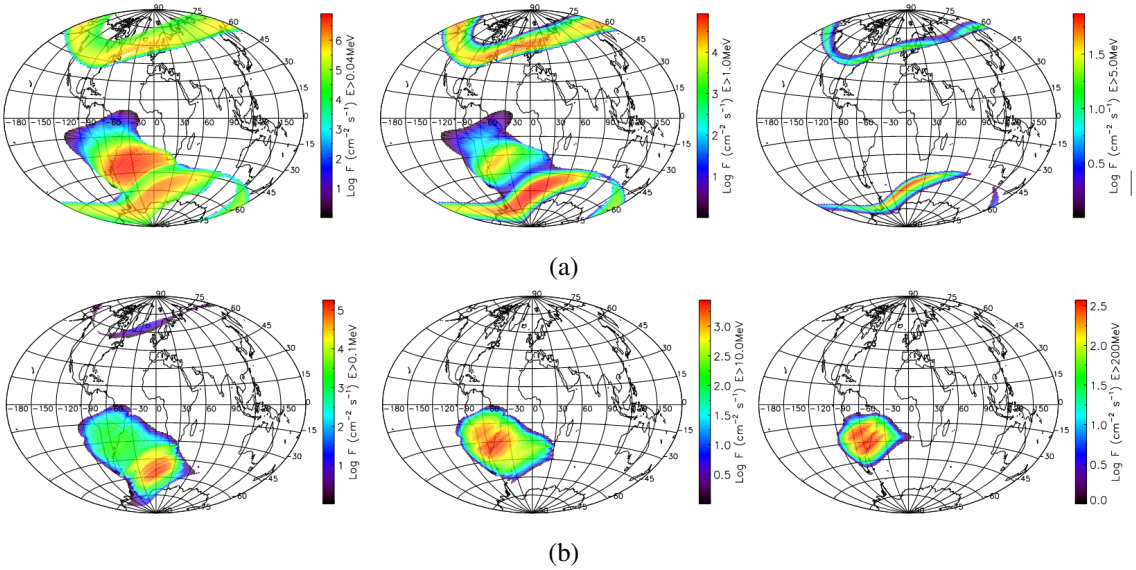


Figure 1.7: **Top panels:** Integral flux maps of trapped electrons for AE8 MAX model at 550 km altitude with low energy thresholds of 40 keV, 1 MeV, and 5 MeV, respectively, from left to right. **Bottom panels:** Integral flux maps of trapped protons for AP8 MIN model at 550 km altitude with low energy thresholds of 0.1 MeV, 10 MeV, and 200 MeV, respectively from left to right (Ripa et al., 2020).

Gamma-ray telescopes use a variety of techniques to remove background noise from their measurements. One common approach is to use a combination of active and passive shielding to reduce the impact of cosmic rays and other sources of background radiation by using a combination of plastic scintillators and lead shields. The BATSE team used Polynomial Fitting Technique, which does cubic fit on $\sim$ 1000 – 2000 s before and after the burst (Band et al., 1993). Non-parametric methods, such as sliding window techniques or rolling medians, can also be employed for background estimation. These methods involve calculating local statistics, such as the mean or median, within a sliding window along the light curve. The resulting values provide an estimate of the background level, which can be subtracted to isolate the signal. These methods are relatively straightforward and are commonly used when the background is smoothly varying (Leon-Anaya et al., 2023). The simplest detection method involves simultaneously monitoring multiple count rates on different detectors or channels, triggering an alert only when the threshold is exceeded on multiple channels.

1.3 Telescopes and instruments

The beginning of GRB detection happened coincidentally when Vela satellites in the late 1960s aimed to detect nuclear explosions but detected emissions that did not originate in Earth's vicinity. Due to the limitations of their equipment and the fact that gamma rays cannot be collimated due to their high energy and penetrating nature, studying and observing gamma-ray bursts present significant challenges. Notable progress in their understanding was made through the implementation of coded masks, which is a patterned array of opaque and transparent elements that are placed in front of a gamma-ray detector. This technique has been successfully employed by space-based observatories such as *Swift* and *INTEGRAL*. When a GRB occurs, the gamma rays interact with the mask, creating a unique shadow pattern on the detector. By analyzing this pattern, scientists can determine the direction, intensity, and spectral properties of the burst. This provided the ability to make follow-up observations and detect afterglows.

1.3.1 Notable missions

VELA was a series of satellites launched by the United States in the late 1960s and early 1970s to monitor compliance with the Limited Test Ban Treaty. While the primary purpose of the VELA satellites was to detect nuclear explosions in space, they also played a significant role in the study of GRBs (Singer, 1965). In fact, the discovery of GRBs can be traced back to VELA satellites. In 1967, the VELA 4 satellite detected an unusual burst of gamma radiation that was initially thought to be a nuclear test by the Soviet Union. However, subsequent observations by other VELA satellites showed that these bursts were not related to nuclear explosions but were instead coming from deep space. These bursts had a duration of just a few milliseconds to several minutes and were found to be coming from random directions in the sky. These observations led to the recognition of gamma-ray bursts as a new class of cosmic phenomena. The VELA satellites detected a total of 16 gamma-ray bursts during their operational lifetime, which ended in the mid-1970s. While these detections were limited in number and duration compared to later missions, they provided the first evidence of the existence of GRBs and set the stage for further research in this field (Klebesadel et al., 1973).

Burst and Transient Source Experiment (BATSE) was a scientific instrument on board the Compton Gamma-Ray Observatory (CGRO) satellite, which was launched by NASA in 1991 and operated until 2000. The primary goal of BATSE was to study GRBs and detect thousands of such bursts during its operation, greatly increasing our understanding of these phenomena. In addition to studying GRBs, BATSE also detected other types of transient events, such as solar flares and soft gamma-ray repeaters. It consisted of eight detectors, each of which could detect gamma rays in the energy range of 20 keV to 8 MeV. Thanks to this instrument, we were able to resolve the extragalactic origin (Figure 1.1) and identify the bimodal character of GRBs (Figure 1.2) (Fishman et al., 1982; Preece et al., 2000).

BeppoSAX was a satellite observatory launched in 1996 by the Italian Space Agency (ASI) in collaboration with the Netherlands Agency for Aerospace Programs (NIVR) (Boella, G. et al., 1997). It was primarily designed to study GRBs and the X-ray universe. Its instruments included the Low-Energy Concentrator Spectrometer (LECS) for $0.1 - 10$ keV X-rays, the Medium-Energy Concentrator Spectrometer (MECS) for $1.3 - 10$ keV X-rays, the High-Pressure Gas Scintillation Proportional Counter (HPGSPC) for $3 - 120$ keV X-rays, the Phoswich Detection System (PDS) for $15 - 300$ keV X-rays, and the Gamma-Ray Burst Monitor (GRBM) for detecting and localizing gamma-ray bursts. One of its major achievements was the discovery of the X-ray afterglow of GRB 970508, which enabled the precise determination of their positions and facilitated the optical follow-up observations that led to the identification of their host galaxies (Bloom et al., 1999).

High Energy Transient Explorer 2 (HETE-2) was a space observatory launched in 2000 with the primary mission of detecting and studying GRBs. HETE-2 primarily aimed to conduct the first multi-wavelength study of GRB using UV, X-ray, and gamma-ray instruments all mounted on a single spacecraft. What set the HETE mission apart was its ability to rapidly and accurately locate GRBs, with a precision of around 10 arcseconds, and quickly transmit that information to ground-based observatories for follow-up studies in radio, IR, and visible light bands. With these capabilities, HETE-2 aimed to shed light on the origin and nature of GRBs (Ricker et al., 2003).

The **Neil Gehrels Swift Observatory** (previously called the Swift Gamma-Ray Burst Explorer) is a space-based observatory that was launched by NASA in November 2004 (Gehrels, 2004). Swift consists of three instruments: the Burst Alert Telescope (BAT) covering an energy range of 15-150 keV; the X-Ray Telescope (XRT) spanning the energy range of 0.2-10 keV; and the Ultraviolet/Optical Telescope (UVOT). The BAT is responsible for detecting and localizing gamma-ray bursts, while the XRT and UVOT observe the afterglows of these events in X-ray and UV/optical wavelengths, respectively. One of the key features of Swift is its ability to quickly and accurately pinpoint the location of GRBs. This allows us to find the source galaxy and detect the afterglow

in various wavelengths. The Swift telescope has made several significant discoveries, including detecting and studying the most distant gamma-ray burst ever observed, which provided insights into the early universe and the formation of the first stars and galaxies (Tanvir et al., 2009). Additionally, the telescope has discovered a new class of gamma-ray bursts, known as "ultra-long" GRBs, which are thought to be produced by the collapse of massive stars or the merger of neutron stars (Gendre et al., 2013).

The **Fermi Gamma-ray Space Telescope** is a space observatory designed to study high-energy gamma rays. The Fermi telescope was launched on June 11, 2008, by NASA (Atwood et al., 2009). The telescope is equipped with two main instruments, the Large Area Telescope (LAT), the detection range of 10 keV – 25 MeV, and the Gamma-ray Burst Monitor (GBM), with a range of 20 MeV – 300 GeV. The Fermi telescope has made many important discoveries since its launch (Ajello et al., 2021): detecting GRB 130427A, which had the largest fluence, highest-energy photon (95 GeV), longest γ-ray duration (20 hours), and one of the largest isotropic energy releases ever observed from a GRB and many detections admirable of gamma-ray bursts (Goldstein et al., 2017).

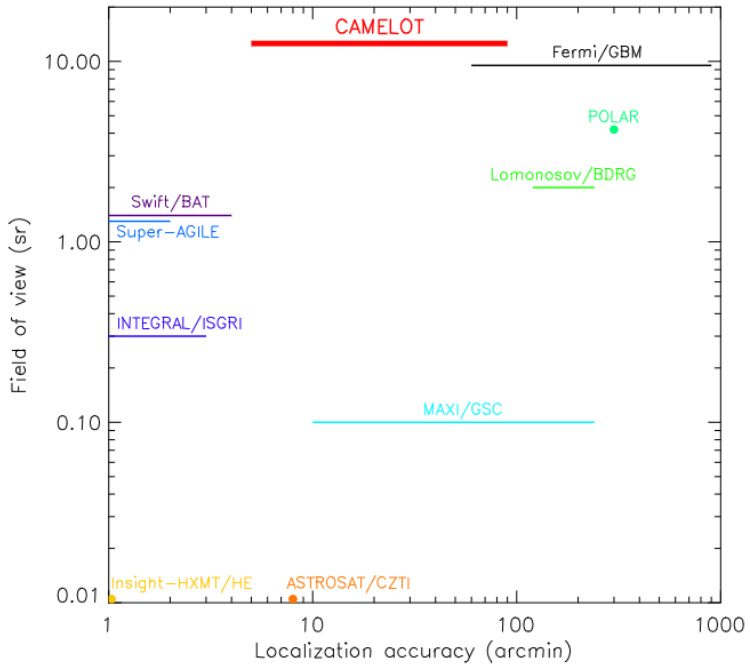


Figure 1.8: Field of view vs localization accuracy for the currently operating GRB monitoring instruments. By providing both all-sky coverage and good localization accuracy, the proposed *CAMELOT* mission fills an empty region in the parameter space (Werner et al., 2018).

The **International Gamma-Ray Astrophysics Laboratory** (INTEGRAL) is a space observatory that was launched on October 17, 2002, by the European Space Agency (ESA) in cooperation with Russia (PROTON launcher) and the United States (Deep Space Network ground station). INTEGRAL is equipped with four instruments: the Imager on Board the INTEGRAL Satellite (IBIS), the Spectrometer on INTEGRAL (SPI), the Joint European X-Ray Monitor (JEM-X), and the Optical Monitoring Camera (OMC). Altogether monitoring gamma-ray in 15 keV – 10 MeV and X-rays in 3 – 35 keV energy range with fine spectroscopy with 12 arcmin FWHM (Winkler et al., 2003). In addition to detecting and studying individual GRBs, INTEGRAL has also contributed to our understanding of the overall GRB population. For example, INTEGRAL's observations have helped to constrain the luminosity function of GRBs (Salvaterra et al., 2008) and the rate of long

GRBs in the local universe (Pescalli et al., 2016).

INTEGRAL, Fermi, and Swift are incredibly valuable missions that have helped astronomers and astrophysicists to observe the high-energy sky. However, they are also very large and expensive, with costs ranging from hundreds of millions to billions of euros. Given their high cost, building and launching these telescopes is a significant challenge, but the demand for observing high-energy spectra continues to be high. To address this issue, a new project called CAMELOT proposes a different approach: using small, inexpensive satellites with the technology called CubeSats.

1.3.2 CubeSats

CubeSat is a type of miniature satellite that is designed to be low-cost and easily deployable. CubeSats are designed to be modular, which means that different components can be added or removed depending on the mission requirements. The standard CubeSat size is called 1U, which is a cube that measures $10 \times 10 \times 10$ cm and weighs about 1 kilogram. However, larger sizes such as 2U, 3U, 6U, and even 12U are also available, which allows for greater flexibility in mission design. They can be launched as secondary payloads on larger rockets, reducing launch costs and making space exploration more accessible to a wider range of organizations. CubeSats also enable faster design and development cycles, allowing researchers to iterate and test new ideas more quickly than with traditional satellite systems. They are now widely used in various applications, including Earth observation, telecommunications, and detecting GRBs.

The **Cubesats Applied for MEasuring and LOcalising Transient** (CAMELOT) project proposes using a fleet of at least nine 3U cubesats equipped with CsI(Tl) scintillator-based soft gamma-ray detectors to perform all-sky monitoring and timing-based localization of gamma-ray transients (Werner et al., 2018). This project is being developed by a team of researchers from several institutions, including Masaryk University. The fleet will be able to determine the source position of bright short gamma-ray bursts with an accuracy of approximately $\sim$ 1 degree by cross-correlating their light curves taken from individual fleet members (Ohno et al., 2020). To achieve this, precise time synchronization and accurate time stamping of detected gamma-ray photons will be achieved by using onboard GPS receivers. The fleet will also have the capability for fast, nearly simultaneous downlink of data using a global inter-satellite communication network. Overall, the proposed fleet is expected to outperform all GRB monitoring missions in terms of all-sky coverage.

A pathfinder project is already ongoing. GRBAlpha and VZLUSAT-2, satellites that carry scaled-down gamma-ray detectors as the ones proposed for the CAMELOT project, more about that in next Chapter 2.1.

[6] Image taken from www.vzlusat2.cz.

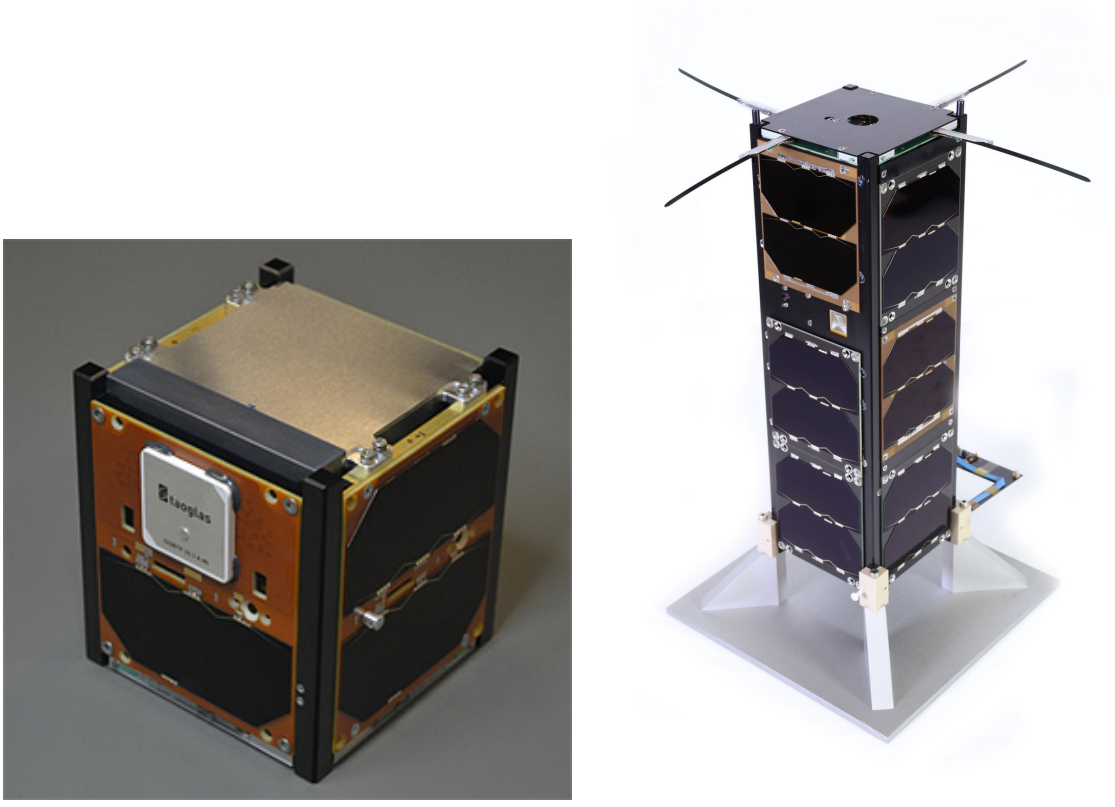


Figure 1.9: **Left:** GRBAlpha (Pál et al., 2020); **Right:** VZLUSAT-2 [6].

Chapter 2

Data

All the data used for this thesis is from GRBAlpha, which I was fortunate to get access to and work with. Due to imbalanced data, having a plethora of data points containing background radiation (Table 2.2), and only few GRB detections (Table 2.4) to have a good sample, I incorporated synthesized data alongside the original GRBAlpha dataset to increase the diversity of the dataset and capture a wider range of scenarios.

2.1 GRBAlpha

GRBAlpha (Pál et al., 2023), launched on 2021 March 22, is a 1U CubeSat mission and its purpose is to serve as an in-orbit demonstration for the detector system on the CAMELOT mission (Figure 1.3). Although GRBAlpha only provides 1/8th of the expected effective area of CAMELOT, the purpose is to validate the core idea of CAMELOT, which is the feasibility of timing-based localization, by comparing the observed light curves with those of other existing GRB monitoring satellites. Since the launch and at the time of writing, GRBAlpha has detected and characterized 23 confirmed GRBs, 9 solar flares, 2 soft gamma repeaters (SGRs) and one X-ray binary outburst, including prominent events like GRB 221009A (Ripa et al., 2023).

Together with Thallium-doped Caesium Iodide CsI(Tl) scintillator and dual-channel multi-pixel photon counter (MPPC), to detect GRBs, on board are GPS receiver, sun-sensor, magnetometers, gyroscopes, and thermometers. GRBAlpha doesn't have an active attitude control system, but it utilizes permanent magnets and magnetically soft material to achieve passive attitude stabilization. Additionally, it obtains attitude information by using MEMS gyroscopes, magnetometers, and sun sensors simultaneously (Pál et al., 2020).

2.1.1 Detection principle

When gamma rays interact with matter, they can transfer their energy to charged particles through interaction mechanisms like the photoelectric effect, Compton scattering, and pair production. When a charged high-energy particle interacts with a scintillator, it causes caesium and iodine atoms to absorb its energy. Consequently, electrons in these atoms move from the valence band, where they leave a positively charged hole, to the conduction band, where they can move freely within the crystal lattice. Thallium atoms are then ionized by these holes, attracting free electrons in the lattice. If an electron that is captured creates an excited configuration with a thallium atom, it drops to the ground state immediately while emitting a photon. Although this deexcitation could take place in a pure crystal, the emitted photons would be too energetic. Therefore, an activator such as thallium is added to the crystal to create its energy states within the lattice, enabling the emission of visible photons. These photons are then detected by arrays of thousands of avalanche photodiodes connected in parallel, known as MPPCs or silicon photomultipliers. As the photons collide with silicon atoms, electron-hole pairs are formed and accelerated by the strong electric field inside the photodiodes, producing secondary electron-hole pairs. This chain reaction continues, ultimately amplifying the output current. In an ideal scenario, each photodiode should be struck by

a maximum of one photon. Consequently, the number of photodiodes that contribute to the output signal, and therefore its amplitude, is directly proportional to the number of scintillation photons (Knoll, 2010).

The output current is directed through the pre-amplifier and shaping amplifier, where it is transformed into a voltage signal and reduced of any electrical noise to determine the pulse height accurately. The pulse height corresponds to the signal generated in the MPPC and, therefore, to the energy of the incident photons. The processed signal then enters the analog-to-digital converter (ADC) and is converted into a digital form. Table 2.1 displays the approximate energies for energy bands utilized in the analysis performed in this work. The conversion between the spectral channel and energy was obtained through pre-launch calibration utilizing various radioactive isotopes (Torigoe et al., 2019), and the resulting conversion is as follows:

$$E\,[\mathrm{keV}] = 4.08 \times \text{ADC channel number} - 154.$$

ADC channel	E [keV]
64 - 128	~ 110 to ~ 370
128 - 192	~ 370 to ~ 630
192 - 256	~ 630 to ~ 890

Table 2.1: The conversion between energy and ADC channel for energy bands used in the analysis.

2.2 Data Selection and Acquisition

Unlike other telescopes dedicated to observing GRBs, GRBAlpha does not have an onboard trigger system to automatically detect and download data. Instead, the data must be manually downloaded based on the confirmed detections of other telescopes. I have chosen to work with GRBAlpha because colleagues from my university are involved in the satellite's operation, and I have direct contact with the main scientists working on the project. This connection provides me with unique opportunities to learn and contribute to this project, making it an exciting and rewarding experience.

The data for this work were taken between March 14th, 2021, and May 15th, 2023. From May 2021 to June 2022, GRBAlpha made a 60 s long observation with full spectral resolution (256 bins) almost every day; since then, only once a week. The remaining observations are usually done in four spectral bands and with an exposure time of 4 s, with a few that were done with 1 s and 60 s exposures. To characterize the long-term degradation of the detector, measurements with high spectral resolution (64 and 256 spectral bins) are done but contain only a few data points and were not used in this thesis. The lowest energy band, ADC channel 0 to 64, has an instrumental noise peak starting around 45 ADC but gradually evolved to 54 ADC and was not used either. Also, six sets of measurements were dropped due to bad time synchronization. Furthermore, 6120 data points contain zero counts, which originated from internal software problems and therefore were dropped. Due to the loss of the VHF radio and similar technical issues, there is no data, with an apparent gap in data (Figure 2.3)

t_{exp}[s]	Data points
1	2003034
2	7985
3	292797
4	361739
15	9224
40	131
60	6

Table 2.2: The number of GRBAlpha data points for each exposure time.

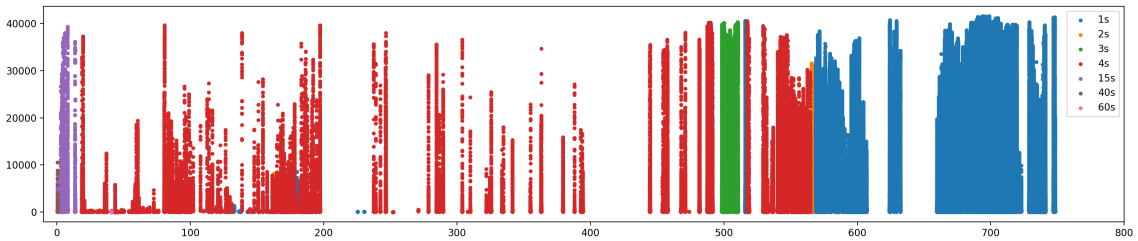


Figure 2.1: Data from GRBAlpha (blue) and color representation of bin sizes from March 14th, 2021, to May 15th, 2023. The x-axis represents time converted to Julian Date and subtracted by the date of the first light (JD - 2459319).

In the data (Figure 2.1), there are three apparent distinct areas with higher background count rates repeating every orbit. Two of them are around the poles, between approximately ± 40 and ± 80 degrees of latitude, depending on the exact longitude. These are regions where the outer Van Allen radiation belt gets closest to the Earth, those are the northern and southern polar rings. The third region is the South Atlantic Anomaly (SAA), extending from approximately 270 to 30 degrees of longitude and from 0 to −60 degrees of latitude, but its shape is very irregular. The SAA is an area over the South Atlantic Ocean where the Earth's magnetic field is weaker, and it allows charged particles from space, including those from the inner Van Allen Belt, to penetrate closer to the Earth's surface.

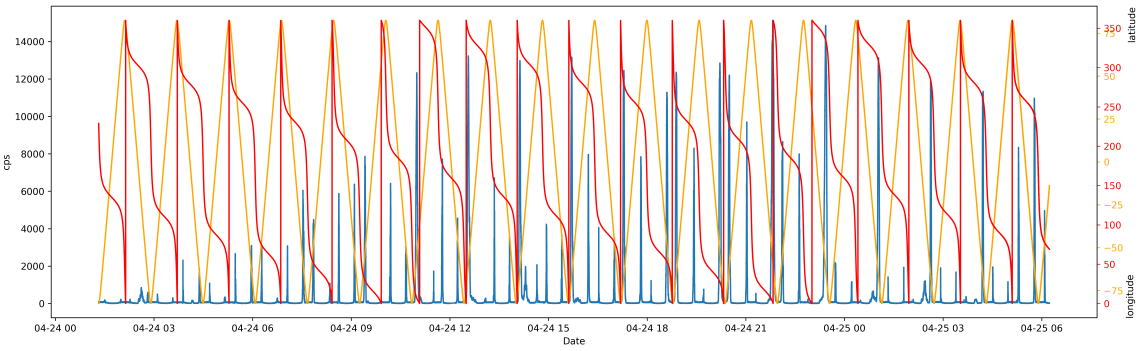


Figure 2.2: Sample of 19 orbits of GRBAlpha.

2.3 Simulated data

Due to a small number of detected GRBs or any non-background high activity, I have decided to simulate additional data for training the models. Creating a large number of simulated events

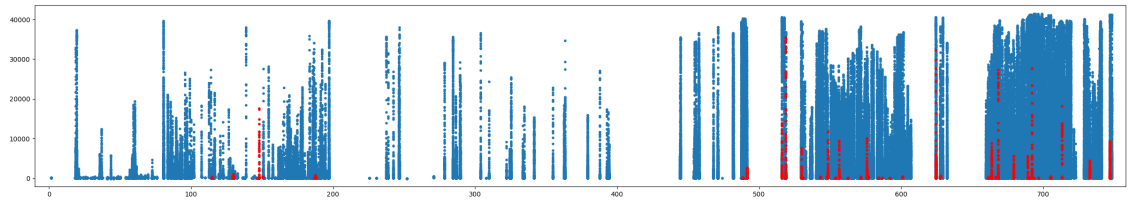


Figure 2.3: Data from GRBAlpha (blue) and confirmed GRB detections (red) from March 14th, 2021, to May 15th, 2023. The x-axis represents time converted to Julian Date and subtracted by the date of the first light (JD - 2459319).

based on the known properties of GRBs and the instrument's response will populate the parametric space, ensuring a more robust and comprehensive dataset for model training.

No two bursts have ever been found to have exactly the same temporal and spectral evolution (Nemiroff et al., 1993). So when simulating GRBs, the main focus was on the most common shape, that being a Fast Rise Exponential Decay (FRED) profile (Figure 1.4). The FRED profile has been found to be a reasonable approximation for many observed GRB light curves. Although not all GRBs follow this exact profile, a significant number of them exhibit a similar shape. It requires only a small number of parameters to describe the shape of the light curve, making it easier to simulate and still capture the general shape.

FRED profile can be written as (Norris et al., 1996):

$$y(t) = A \cdot \exp\left\{-\left(\frac{|t - t_{\max}|}{\sigma_r}\right)^{\nu}\right\}; \quad t \leq t_{\max} \tag{2.1a}$$

$$y(t) = A \cdot \exp\left\{-\left(\frac{|t - t_{\max}|}{\sigma_d}\right)^{\nu}\right\}; \quad t > t_{\max}, \tag{2.1b}$$

where $t_{\max}$ is the time of the pulse's maximum intensity, A; σ_r and σ_d are the rise ($t \leq t_{\max}$) and decay ($t > t_{\max}$) time constants, respectively; and ν is a measure of pulse sharpness ("peakedness", lower number imply a more peaked pulse).

In order to construct these curves, a 64-bin window with 1 s time-bins was selected. This choice was made as a balanced compromise between capturing both the entirety of the burst and a portion of the background while also considering the model's complexity and efficiency. The range of $t_{\max}$ spanned from 8 to 56 bins, ensuring that either the entire burst or the majority of it was within the window. For $\nu = 1$ or 2, Equation 2.1 describes a two-sided exponential or Gaussian, respectively. Figure 2.4 illustrates pulse shapes, with $\sigma_d/\sigma_r = 2.5$, near the most frequently occurring decay-to-rise ratio. The most frequently occurring peakedness lies approximately halfway between Gaussian and exponential, but higher values of $\nu(\gtrsim 4)$, which result in flat-topped pulses, often occur too, but are mainly the result of low temporal resolution or broad overlapping structures, so chosen range is from 1 (spikes) to 5 (FRED). As mentioned, the most frequent decay-to-rise ratio is 2.5. Decay time σ_d is selected based on the anticipated time range of GRBs, which is calculated as $\tau_{t,d} = [\ln(2)]^{1/\nu}\sigma_{r,d}$. The time range is determined by the temporal resolution (1s) and the time window utilized (64 s), with the additional constraint that the peak position must fall between the 8th and 56th bin. This translates to a range of 2 to 48 seconds, corresponding to σ_d values ranging from 1 to 26. The amplitude was calculated as signal-to-noise ratio (SNR) times mean background level, which gives us more robust control about its peak amplitude than the count rate range. SNR range was determined by analyzing already detected GRB from GRBAlpha (Table 2.4), GRB 221127A having the smallest SNR 3.0, and if we do not count the BOAT (GRB 230307A) with

305.1 (Figure 1.6), the highest is GRB 230522A with 47.0. So the SNR was log-uniformly selected in the range 3 to 25 because anything above SNR 25 is significant and detectable with a simple threshold method, and we are more interested in finding lower SNR GRBs.

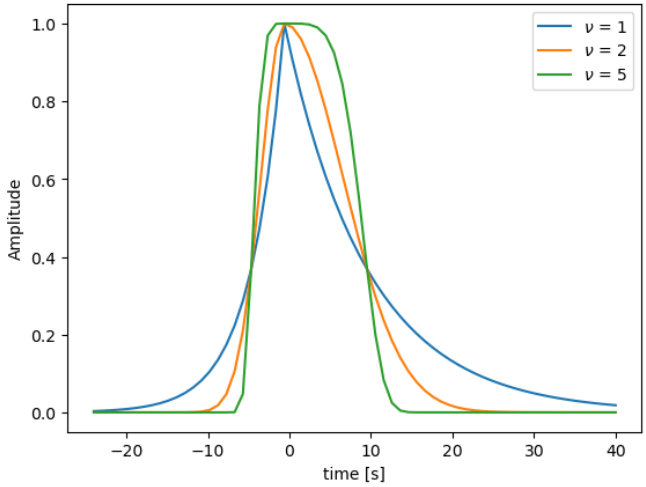


Figure 2.4: Pulses with different ν ($\sigma_d/\sigma_r = 2.5$).

As mentioned, the generation of the GRB peak is also based on the background level. Since our goal is to identify GRBs occurring in lower background regions, it is essential to locate these areas and determine their respective levels. Figure 2.5 shows a logarithmic histogram of GRBAlpha's background levels. We see two main peaks around 75 and 150 counts, with the end of the tail around 250, which corresponds with regions between north and south polar rings and region within polar rings (Figure 1.7), respectively, which you can see in Figure 2.2. Only these two low-background regions are of interest because, in high-background regions, it is difficult or nearly impossible to find GRBs. So the range for the constant background was uniformly set between 75 to 250. Because the high and low background regions do not have strict edges, a linear trend with a slope range between -1.5 and 1.5 was added. The distribution of simulated windows with the constant and linear background was split evenly with a relative frequency of 50% for both types.

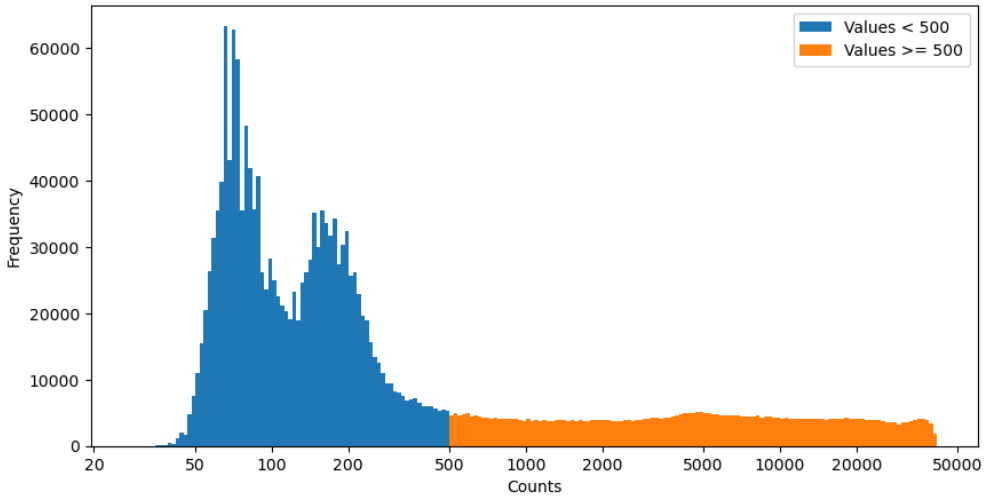


Figure 2.5: Histogram showing the frequency distribution of log-transformed counts for all background data, with bins colored in blue for values less than 500 and orange for values equal to or greater than 500 and peaking at 65 counts.

chunk ID	data ID	exp start time	exp end time	sec from 1st exp start	expo-sure	spec nbins	n chunks
0	27540	2022-11-12 16:20:59	2022-11-12 16:21:00	0	1	4	9309
1	27539	2022-11-12 16:21:00	2022-11-12 16:21:01	1	1	4	9309
2	27538	2022-11-12 16:21:01	2022-11-12 16:21:02	2	1	4	9309
3	27537	2022-11-12 16:21:02	2022-11-12 16:21:03	3	1	4	9309
4	27536	2022-11-12 16:21:03	2022-11-12 16:21:04	4	1	4	9309

(a) Time and bin information.

lon start	lat start	alt start	lon end	lat end	alt end
314.355	-78.0328	565.987	314.163	-77.9842	565.964
314.163	-77.9842	565.964	313.973	-77.9354	565.941
313.973	-77.9354	565.941	313.784	-77.8866	565.917
313.784	-77.8866	565.917	313.596	-77.8376	565.894
313.596	-77.8376	565.894	313.410	-77.7885	565.871

(b) Spatial information.

temp1(° C)	temp2(° C)	temp3(° C)
2.53000	2.69000	2.88000
NaN	NaN	NaN
NaN	NaN	NaN
NaN	NaN	NaN
NaN	NaN	NaN

(c) Temperature information, NaN present due to temperature information is recorded only every minute.

cnt band0	cnt band1	cnt band2	cnt band3	cps band0	cps band1	cps band2	cps band3	cnt sum	cps sum
1683	2404	480	92	1683	2404	480	92	4659	4659
1670	2482	524	143	1670	2482	524	143	4819	4819
1648	2580	547	127	1648	2580	547	127	4902	4902
1762	2769	512	140	1762	2769	512	140	5183	5183
1777	2948	627	160	1777	2948	627	160	5512	5512

(d) Counts and count rate in 4 bands (2.1).

Table 2.3: The example format of five raw data points from November 12th, 2022

2.3.1 Augmented peaks

In order to enhance the performance of the transient prediction model, we investigated the use of Poisson noise augmentation for selected GRB peaks which serve as essential references due to their well-characterized profiles. Adding Poisson noise is beneficial for augmenting the peaks of GRBs for several reasons. Firstly, it enables the realistic simulation of statistical fluctuations and random variations present in GRBs. By incorporating Poisson noise into the peaks, the synthetic data becomes more representative of the natural variations observed in these astronomical events. Secondly, the variability introduced by the Poisson distribution reflects the inherent fluctuations in burst intensities, which are crucial for training models to accurately distinguish genuine peaks from background noise. Finally, as the Poisson distribution is well-suited for modeling rare and random events, it aligns with the characteristics of GRBs and allows the models to learn and adapt to their statistical properties.

The selected criteria for augmenting GRB peaks included: requiring a bin resolution of 1 second, ensuring visual detectability, SNR above 3, and possessing representative (FRED) profiles. In Table 2.4 are highlighted 11 GRBs and their corresponding SNR that were selected, with plots in Appendix (Figure 1).

GRB	SNR
GRB 230522A	47.0
GRB 230512A	11.0
GRB 230510B	15.7
GRB 230510A	6.6
GRB 230328B	**13.3**
GRB 230320B	**22.5**
GRB 230307A	**305.1**
GRB 230305A	**8.3**
GRB 230304B	**28.4**
GRB 230207B	**25.0**
GRB 230102A	**5.6**
GRB 221206B	**35.8**
GRB 221127A	3.0
GRB 221122A	**4.3**
GRB 221119A	**23.5**
GRB 221112A	3.4
GRB 221107A	**9.3**
GRB 221029A	9.8
GRB 221022B	22.8
GRB 221020A	11.3
GRB 220927A	19.7
GRB 220926B	4.8

Table 2.4: All GRB detections of GRBAlpha[1] (between March 14th, 2021, and May 15th, 2023), in bold are GRBs that were picked for augmentation and for testing.

[1] All confirmed detection of GRBAlpha are updated on this site: monoceros.physics.muni.cz/hea/GRBAlpha/.

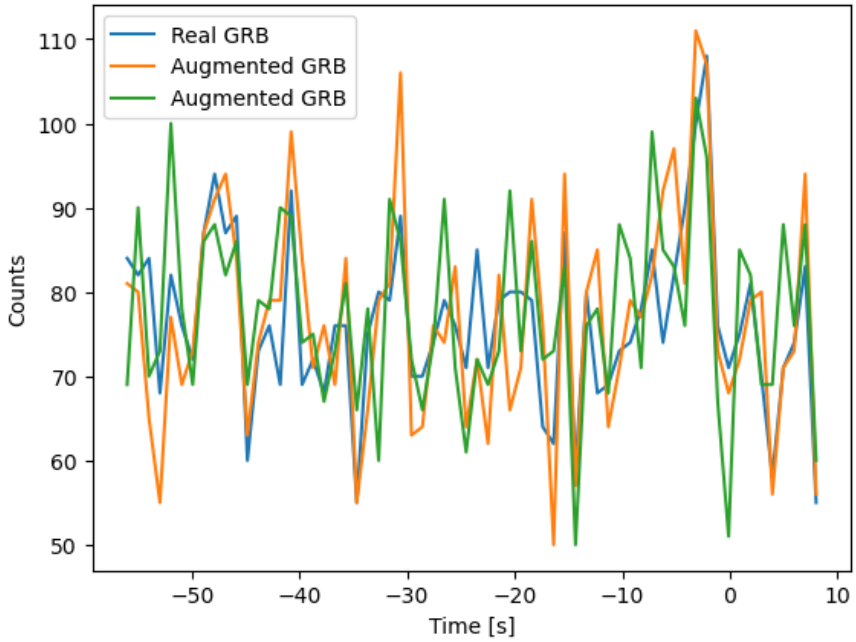


Figure 2.6: Example of a real GRB and its two augmented versions (peak at $t = 0$).

Chapter 3

Machine Learning

Machine learning (ML) is a field of artificial intelligence (AI) that involves developing algorithms and models that enable computers to learn from and make predictions or decisions based on data without being explicitly programmed. Machine learning has various applications in areas like image recognition, language processing, recommendation systems, and more. One of the most popular and well-known applications of ML is Artificial Neural Networks (ANN), which are modelled after the structure and function of the human brain. It consists of interconnected layers of artificial neurons (Equation 3.1), which are connected by weights (Section 3.2.2).

From powering our virtual assistants, optimizing supply chain logistics, medical diagnosis, recommending cat videos, or filtering unwanted email, ML is now everywhere. Astronomy and astrophysics are not behind (Baron, 2019). The rapid development of telescopes and detectors brings us not only new and fascinating discoveries but also enormous quanta of data (Ball & Brunner, 2010; Pesenson et al., 2010). The following is a selection of works that, in my humble opinion, deserve acknowledgement: (i) Classifying galaxies based on their morphology, colour, and other features (Raddick et al., 2010); (ii) Detecting and classifying different types of supernovae based on their light curves and spectra (Foley et al., 2007); (iii) Detecting exoplanets based on their transit signals. This led to the discovery of many new exoplanets, including some that are potentially habitable (McCauliff et al., 2015); (iv) classifying variable starts based on their light curves (Mahabal et al., 2017; Bassi et al., 2021); (v) gamma-ray burst detection (Abraham et al., 2021).

In summary, ML is a valuable tool for several reasons, including its ability to handle large amounts of data, automate tasks, quickly analyze data to identify patterns, make more accurate predictions and decisions than traditional methods, and learn and adapt to new data.

3.1 Supervised, unsupervised and semi-supervised learning

There are three main types of Machine Learning: Supervised Learning, Unsupervised Learning, and Semi-supervised Learning. The structure of your ML project depends on what kind of data you have available and what kind of information you want to extract from them.

In Supervised Learning, the model is trained on labeled data, which means the dataset contains input features and their corresponding output labels. The goal of the algorithm is to learn the mapping between the input and output variables so that it can make accurate predictions on new, yet unseen data. Examples of Supervised Learning algorithms include Linear Regression, Logistic Regression, Support Vector Machines, and Neural Networks (Liu & Wu, 2012).

Unsupervised Learning, on the other hand, involves training the model on unlabeled data. In this type of learning, the model tries to identify patterns or groupings in the data without being given any prior information. This approach is commonly used for clustering and anomaly detection (Mohan, 2017).

Semi-supervised learning is a type of Machine Learning that lies somewhere between Supervised Learning and Unsupervised Learning. In this type of learning, the model is trained on a

combination of labeled and unlabeled data. The labeled data is used to train the model initially, while the unlabeled data is used to refine the model's predictions over time. By combining the labeled and unlabeled data, semi-supervised learning algorithms can often achieve better results than using either type of data alone. Semi-supervised learning is commonly used in applications where labeled data is scarce, such as in Natural Language Processing or Speech Recognition (Getz et al., 2006).

3.2 Building blocks of neural networks

As said at the beginning of the chapter, neural networks are a type of machine learning algorithm that is loosely modelled on the structure and function of the human brain. The human brain operates in a forward manner, processing information through interconnected neurons and adapting through synaptic plasticity, which refers to the brain's ability to strengthen or weaken connections between neurons based on their activity and experiences. Artificial neural networks also have a forward flow through interconnected layers of artificial neurons, but use backpropagation (Section 3.3.4) to adjust weights and improve performance (Rumelhart et al., 1986). The upcoming subsection will offer a comprehensive overview of what neural networks are made of.

3.2.1 Artificial neuron

The terms "artificial neuron" and "perceptron" are often used interchangeably to refer to the same thing: a basic unit of a neural network. It takes one or more input values, multiplies them by weight, adds them together, and passes the result through an activation function. However, there is a historical difference between the two terms.

The term "artificial neuron" was first introduced in the 1940s by Warren McCulloch and Walter Pitts (McCulloh & Pitts, 1943). They proposed a model of an artificial neuron that could perform simple logical operations, such as AND and OR. The term "perceptron", on the other hand, was introduced in the late 1950s by Frank Rosenblatt (Rosenblatt, 1958). Rosenblatt's perceptron was an artificial neuron that could learn to classify input data into two categories (i.e., binary classification). It used a simple learning algorithm called the "perceptron learning rule" to adjust its weights and improve its classification accuracy over time. The perceptron was one of the first successful Machine Learning models, laying the groundwork for developing more complex neural network architectures.

Mathematically, the perceptron can be described as follows: input values $x_1, x_2, ..., x_n$ and weights $w_1, w_2, ..., w_n$ with bias w_0 are then passed through an activation function f to produce a final output y:

$$y = f\left(\sum_{n=1}^{N} w_0 + w_n x_n\right). \tag{3.1}$$

Neurons are organized into layers. The input layer receives the initial data, the hidden layers process and transform it, and the output layer generates the final prediction. Hidden layers enable the network to learn and extract complex features and patterns from the input data through their interconnected computations.

3.2.2 Weights and bias

Weights represent trainable parameters of the neural network that determine how the information flows through the network. Each connection between two neurons in the network is associated with a weight, which represents the strength of the connection. However, this weighted sum alone

may not be enough to produce the desired output for a given set of inputs, the neuron would only produce an output of 0 when all of its inputs are 0. An additional parameter **bias** is added to the weighted sum of the inputs in each neuron to help shift the output of the neuron, allowing it to produce a broader range of outputs. So even when all of its inputs are 0, the neuron can produce a non-zero output.

3.2.3 Activation function

Activation functions play a crucial role in the functioning of artificial neural networks (ANNs) by introducing non-linearity to the network's decision-making process. The choice of activation functions behaves differently in the hidden layers and output layers. Some commonly used activation functions include:

The **Hyperbolic Tangent** (Tanh) maps inputs to a range between -1 and 1. In hidden layers, tanh functions were widely used before the emergence of ReLU, which will be explained next. In the output, layers are used similarly to the sigmoid function in binary classification (Anireh & Osegi, 2016).

$$f(z) = \frac{e^{2z} - 1}{e^{2z} + 1}.$$

The **Rectified Linear Unit** (ReLU) function is a simple threshold function that outputs the input value if it is positive, and 0 if it is negative. It is one of the most commonly used activation functions in deep learning (Nair & Hinton, 2010; Agarap, 2019). However, ReLU has one limitation. When the input is negative, the gradient becomes zero, leading to a phenomenon called "dying ReLU." This issue can cause neurons to become inactive and cease learning, making them unable to recover during training.

$$f(z) = \max(0, z)$$

Leaky ReLU is an improved version of ReLU that addresses the dying ReLU problem. It introduces a small slope a for negative values instead of completely turning them off (Dubey & Jain, 2019).

$$f(z) = \max(az, z)$$

The **Sigmoid** function maps any input value to a value between 0 and 1. In hidden layers, sigmoid functions were commonly used in the past, but due to causing the vanishing gradient problem 3.2.3, more efficient functions are used (ReLU). In the output layer, sigmoid functions are still used for binary classification, as they can provide a probabilistic interpretation of the prediction (Pratiwi et al., 2020).

$$f(z) = \frac{1}{1 + e^{-z}}$$

The **Softmax** activation function is commonly used in the output layer of multi-class classification tasks. It normalizes the outputs of neurons in the layer, converting them into probabilities that sum up to 1. Softmax ensures that the predicted probabilities represent the relative likelihoods of different classes (Pearce et al., 2021):

$$f(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

.

where $z = \mathbf{wx} + b$, $\mathbf{w}$ is a matrix of weights, b is bias and $\mathbf{x}$ is input data. The linear activation function is also used, typically in the output layer of a neural network when the task at hand involves regression, where the goal is to predict a continuous value.

When using activation functions with small gradients, such as sigmoid and tanh, the gradients can become very small as they propagate backwards through the network, resulting in very slow convergence or even convergence to suboptimal solutions. This is called the **vanishing gradient problem** , which is a well-known issue in deep learning, particularly in the training of deep neural networks with many layers. Several techniques have been developed to address the vanishing gradient problem. One approach is to use activation functions with larger gradients, such as ReLU and its variants, which help to maintain a consistent gradient throughout the network. Or try to use an alternative optimization algorithm, such as the Adam optimizer (Section 3.3.3), which adaptively adjusts the learning rate for each weight based on its historical gradients.

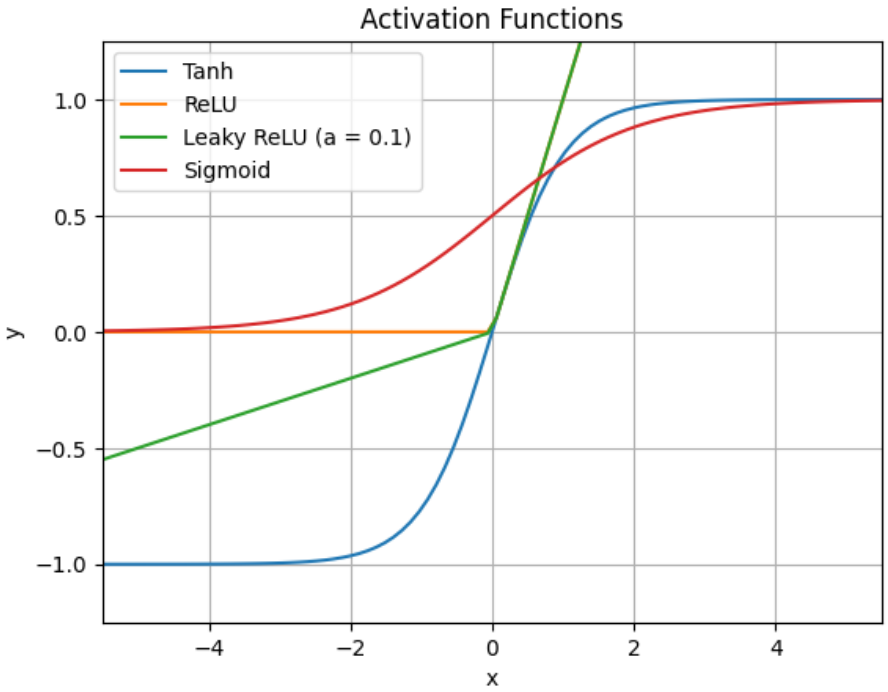


Figure 3.1: Graphical representation of selected activation functions.

3.2.4 Batch normalization

Training ANN can be affected by internal covariance shift that refers to the change in the distribution of layer inputs as the parameters of preceding layers are updated during the training process of a neural network. It occurs because the distribution of inputs to each layer is highly dependent on the parameters of the preceding layers. As these parameters change during training, the distribution of inputs also changes, leading to the internal covariance shift. This phenomenon can slow down the training process and make it harder for subsequent layers to learn effectively. Batch Normalization is a technique used to mitigate this issue by normalizing the inputs of each layer. It operates on a mini-batch of training examples and adjusts the activation functions by subtracting the batch mean and dividing by the batch standard deviation. The normalized activation functions are then scaled and shifted using learned parameters (Ioffe & Szegedy, 2015).

3.2.5 Dropout

The dropout layer is a regularization technique to prevent overfitting and improve generalization. During training, the dropout layer stochastically determines which units to drop out, set to zero, based on a dropout rate, which introduces noise and forces the network to learn redundant representations, making it more robust. This random dropout process is applied independently to each training example, creating an effect similar to training multiple neural networks with different architectures, ultimately lowering variance. During testing, the dropout layer is typically turned off, and the network approximates the average behaviour of the ensemble formed by the sub-networks (Srivastava et al., 2014).

3.3 Training a model

The success of a machine learning model depends on the quality of the data it is trained on. To ensure the model is accurate and effective, we divide the data into training, testing, and validation sets (Section 3.3.1). To evaluate how well the model is able to predict the right output for a given input, the loss function (Section 3.3.2) is computed, so the learning process involves minimizing this function. It also involves finding the right weights and biases (Section 3.2.2) through optimizing techniques (Section 3.3.3).

3.3.1 Training, testing and validating data

The data splitting process involves selecting a portion of the available data to be used for training the model and setting aside the remaining portion for validating during training (hyperparameter search and to ensure the model just does not memorize the data) and testing (how does it perform on new data). The common percentage of splitting datasets can vary depending on the dataset's size, the model's complexity, and the specific task at hand. However, a common practice is to use an $80/20$, where 80% of the data is used for training, and the remaining 20% is used for testing and validation. For smaller datasets, a $50/25/25$ split may be used, where 50% of the data is used for training, and 25% each is used for testing and validation. In some cases, cross-validation techniques may also be used, where the data is split into multiple folds, and the model is trained and tested on different subsets of the data.

Before splitting the dataset, it is essential to shuffle the dataset to ensure that the data is randomly distributed across the training, validation, and testing sets. Without shuffling, the dataset may have a particular order or pattern that could influence the model's training and performance. For example, if the dataset is sorted by class labels or some other feature, the model may learn to rely on this order and perform poorly on unseen data. Shuffling the data ensures that the model is exposed to a random and diverse set of examples during training, making it more robust and able to generalize well to new data. However, shuffling the dataset is not recommended for time-series data, as the order of the samples is essential to model the time-dependent relationships between them. In time-series data, the dataset is typically split using a sliding window approach, where a fixed number of contiguous samples are used to predict the next sample in the sequence. This sliding window approach ensures that the time-dependency structure is maintained, and the model can accurately capture the patterns and relationships between the samples (Nguyen et al., 2021).

3.3.2 Loss function

A loss function is a mathematical function used to evaluate how well a neural network is performing on a given task. The goal of a neural network is to minimize the loss function, which means reducing the difference between the predicted output and the actual output.

There are many different loss functions used in neural networks, and the choice of which one to use depends on the specific task at hand (classification/regression). Here are some common loss functions used in neural networks:

- **Mean Squared Error (MSE)**: The mean squared error loss function is commonly used for regression tasks (Ren et al., 2022). It measures the average squared difference between the predicted and actual output. The formula for MSE is:

$$\text{MSE} = \frac{1}{N} \cdot \sum_{i}^{N} (y_{\text{pred},i} - y_{\text{true},i})^2$$

  where N is the number of samples (batch size), y_{pred} is the predicted output, and y_{true} is the actual output. MSE penalizes large errors more heavily than small errors, which can make it sensitive to outliers.

- **Mean Absolute Error (MAE)**: Mean absolute error loss function is another commonly used regression loss function (Qi et al., 2020; Wang et al., 2023). It measures the average absolute difference between the predicted and actual output. The formula for MAE is:

$$\text{MAE} = \frac{1}{N} \cdot \sum_{i}^{N} |y_{\text{pred},i} - y_{\text{true},i}|$$

  where N is the number of samples, y_{pred} is the predicted output, and y_{true} is the actual output. MAE penalizes large errors less heavily than MSE and can be less sensitive to outliers.

- **Huber Loss**

  Huber loss is a loss function used in regression problems, particularly in cases where the data may contain outliers. The Huber loss combines the properties of both MSE and MAE by using a quadratic loss for small errors and a linear loss for large errors (Gokcesu & Gokcesu, 2021).

$$L(y_{\text{true}}, y_{\text{pred}}) = \begin{cases} 0.5 \cdot (y_{\text{true}} - y_{\text{pred}})^2, & \text{if } |y_{\text{true}} - y_{\text{pred}}| \leq \delta \\ \delta \cdot (|y_{\text{true}} - y_{\text{pred}}| - 0.5 \cdot \delta), & \text{otherwise} \end{cases}$$

  where δ is a threshold parameter that determines the point at which the loss function transitions from quadratic to linear.

- **Binary Cross-Entropy**: Binary cross-entropy loss function is used for binary classification tasks (Ruby & Yendapalli, 2020; Zhang & Sabuncu, 2018; Gordon-Rodriguez et al., 2020). It measures the difference between the predicted probability and the actual label. The formula for binary cross-entropy is:

$$\text{BCE} = -\frac{1}{N} \sum_{i=1}^{N} \left(y_{\text{true},i} \log\left(y_{\text{pred},i}\right) + (1 - y_{\text{true},i}) \log\left(1 - y_{\text{pred},i}\right) \right)$$

  where y_{pred} is the predicted probability and y_{true} is the actual label. BCE penalizes strongly for wrong predictions and tends to give high loss values to predictions that are very wrong.

- **Categorical Cross-Entropy**: Categorical cross-entropy loss function is used for multi-class classification tasks (Li et al., 2022). It measures the difference between the predicted probabilities and the actual label. The formula for categorical cross-entropy is:

$$\text{CCE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{\text{true},i,c}\log\left(y_{\text{pred},i,c}\right)$$

  where y_{pred} is the predicted probability and y_{true} is the actual label, and C is the number of classes. CCE gives high loss values for incorrect predictions and smaller loss values for correct predictions.

The loss function plays a crucial role in training a neural network as it guides the optimization process toward finding the best set of weights that will minimize the difference between the predicted and actual output. An ideal loss function should be differentiable, allowing for gradient-based optimization methods, and have a monotonic increase with error to ensure that the optimization process moves toward minimizing the error.

3.3.3 Optimization algorithm

Optimizing algorithms are an essential component of training neural networks to learn from data. The goal of these algorithms is to find the optimal set of weights and biases for the network that minimizes the loss function on the training data. There are several types of optimizing algorithms, each with its strengths and weaknesses.

Here is a brief overview of some commonly used optimizing algorithms in NNs:

Gradient Descent is the most basic optimization algorithm used in NNs. It works by updating the weights in the opposite direction of the gradient of the loss function. The learning rate parameter controls the size of the weight updates at each iteration (Bryson Jr & Ho, 1969).

Stochastic Gradient Descent (SGD) is a variant of gradient descent that randomly selects a subset of the training data at each iteration to compute the gradient. This helps to avoid getting stuck in local minima and can speed up the convergence of the algorithm (Robbins, 1951).

RMSprop is an adaptive learning rate optimization algorithm that divides the learning rate by the running average of the squared gradient. This helps to reduce the learning rate for weights with high variance in the gradients (Graves, 2013).

Adam is an adaptive learning rate optimization algorithm that combines the ideas of momentum and RMSprop. It keeps track of the first and second moments of the gradients and uses this information to adapt the learning rate for each weight (Kingma & Ba, 2014).

The choice of optimizing algorithm depends on the specific problem and network architecture. Some factors to consider when selecting an algorithm include the size of the dataset, the complexity of the model, and the computational resources available.

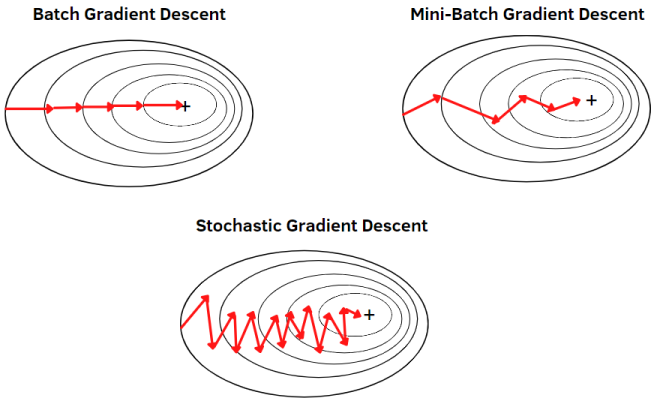


Figure 3.2: Visual representation[1] of finding the minima of loss function by gradient descend approach differing in the amount of data they use in each gradient update.

3.3.4 Backpropagation

Backpropagation, proposed by Rumelhart et al. (1986), is a commonly used algorithm when training neural networks. It involves an efficient way of calculating the gradients of the loss function with respect to the weights of the network and then updating those weights in the direction of the negative gradient. The algorithm proceeds in two stages: a forward pass to calculate the output of the network given the input and a backward pass to calculate the gradients of the loss function.

Let x be the input to the network, y be the desired output, and $f(x; w)$ be the output of the network given input x and weights w. The loss function $L(y, f(x; w))$ measures the difference between the desired output and the actual output of the network. The goal of backpropagation is to update the weights w so as to minimize the loss function.

In the forward pass, the output of each neuron in the network is calculated as follows:

$$z_i = \sum_{j=0}^{n} w_{ij} x_j$$
$$a_i = \sigma(z_i)$$

where z_i is the weighted sum of the inputs to neuron i, w_{i0} is the bias term for neuron i, $F(z_i)$ is the activation function, and n is the number of inputs to neuron i. The output of the network is the output of the final neuron: $f(x; w) = a_n$.

In the backward pass, the gradient of the loss function with respect to the weights is calculated using the chain rule:

$$\begin{aligned}\frac{\partial L}{\partial w_{ij}} &= \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial w_{ij}} \\ &= \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial z_i}\frac{\partial z_i}{\partial w_{ij}} \\ &= \frac{\partial L}{\partial a_i}\sigma'(z_i)x_j\end{aligned}$$

[1] Downloaded from https://www.analyticsvidhya.com/.

where $\sigma'(\cdot)$ is the derivative of the activation function. The gradient of the loss function with respect to the bias term is:

$$\begin{aligned}\frac{\partial L}{\partial w_{i0}} &= \frac{\partial L}{\partial z_i}\frac{\partial z_i}{\partial w_{i0}} \\ &= \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial z_i}\end{aligned}$$

The weights and biases are then updated using the gradients and a learning rate α:

$$\begin{aligned}w_{ij} &\leftarrow w_{ij} - \alpha\frac{\partial L}{\partial w_{ij}} \\ w_{i0} &\leftarrow w_{i0} - \alpha\frac{\partial L}{\partial w_{i0}}\end{aligned}$$

3.3.5 Underfitting and overfitting

Underfitting and overfitting are two common problems in machine learning when training a model. Both issues can lead to poor performance and inaccurate predictions.

Underfitting happens when a model is too simple to capture patterns in the data, leading to poor performance. It can be caused by insufficient complexity, data, or important features, as well as excessive regularization. The result is low accuracy and ineffective predictions.

On the other hand, overfitting occurs when the model is too complex and can perfectly fit the training data, including the noise and outliers. This means the model has too many parameters and can fit the training data too well, resulting in low bias but suffering from high variance. When a model overfits the data, it may perform well on the training set but does not generalize well to new, unseen data, resulting in poor testing accuracy.

To prevent underfitting, the model should be more complex, and additional features can be added to capture the underlying patterns in the data. On the other hand, the model should be simplified to prevent overfitting, and the number of features or parameters should be reduced. Regularization techniques, including L1 and L2 regularization (Mazilu & Iria, 2011; Kolluri et al., 2020) which add penalties on weights, as well as dropout (Section 3.2.5), which randomly deactivates neurons, can effectively prevent overfitting.

Finding the right balance between underfitting and overfitting is crucial to building an accurate and reliable machine-learning model. The model should be complex enough to capture the underlying patterns in the data while being simple enough to generalize well to new data. Cross-validation is a common technique used to evaluate the model's performance and find the optimal balance between underfitting and overfitting.

3.3.6 Accuracy metrics

There are several metrics that can be used to measure the accuracy of machine learning models, depending on the specific task and problem being addressed. One of the most common metrics for classification (Raschka & Mirjalili, 2019) and regression tasks (James et al., 2013):

- Classification Metrics:
  - **Accuracy**:

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

[2] Downloaded from https://www.analyticsvidhya.com/.

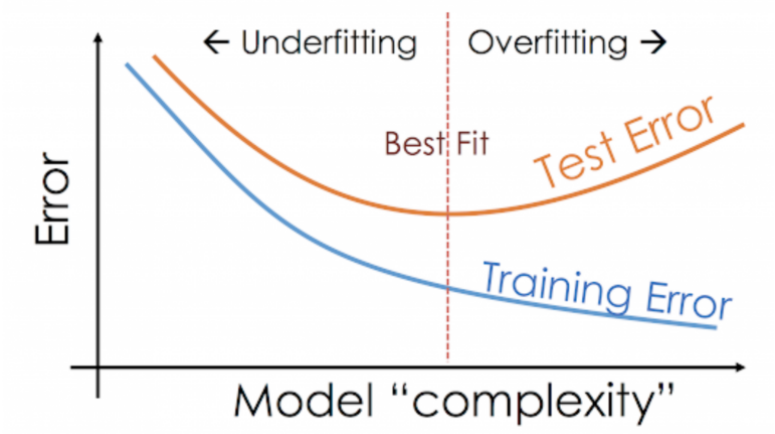


Figure 3.3: Visual representation of finding the balance of underfitting and overfitting[2].

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. Accuracy is commonly used in classification tasks where the goal is to predict the correct class or category.

– **Precision**:

$$\text{PREC} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision measures the proportion of correctly predicted positive instances out of the total predicted positive instances. Precision is useful in situations where the cost of false positives is high. It is commonly used in applications such as fraud detection and medical diagnosis, where false positives can have significant consequences.

– **Recall**, also known as sensitivity or true positive rate (TPR):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Recall measures the proportion of correctly predicted positive instances out of the total actual positive instances. The recall is valuable when the cost of false negatives is high. It is frequently used in tasks such as disease detection, where missing a positive case can have severe implications.

– **F1 Score**:

$$\text{F1 Score} = 2 \times \frac{\text{PREC} \times \text{TPR}}{\text{PREC} + \text{TPR}}$$

The F1 score is the harmonic mean of precision and recall. It provides a balanced measure between the two. The F1 score is widely used when there is an uneven class distribution or an imbalance between the importance of precision and recall. It is a helpful metric in cases that need a balance between identifying relevant instances (recall) and ensuring their correctness (precision), like document classification.

– **ROC AUC**: The ROC AUC (Receiver Operating Characteristic Area Under the Curve) measures the performance of a model in terms of its ability to discriminate between positive and negative classes by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds. The area under the ROC curve (ROC AUC) is a single scalar value that summarizes the overall performance of the model.

  - **Confusion matrix**: The confusion matrix is a grid that organizes predictions made by a model for different classes or categories. In the case of binary classification, the matrix consists of two classes labeled as "positive" and "negative". The numbers within the matrix represent the counts or frequencies of the model's predictions for each class. Specifically, the matrix includes True Positive (TP) for correct positive predictions, True Negative (TN) for correct negative predictions, False Positive (FP) for incorrectly predicted positives, and False Negative (FN) for incorrectly predicted negatives. These values provide valuable insights into the model's performance by illustrating the distribution of accurate and inaccurate predictions with evaluation metrics like accuracy, precision, recall, and F1 score.

- Regression Metrics:

  - **Mean Squared Error** (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{true}} - \hat{y}_{\text{pred}})^2$$

    MSE measures the average squared difference between the predicted values ($\hat{y}_{\text{pred}}$) and the true values (y_{true}) of a regression model, where n is the number of instances. MSE is used where the goal is to predict a continuous numerical value. It quantifies the overall magnitude of prediction errors, giving more weight to larger errors.

  - **Mean Absolute Error** (MAE):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_{\text{true}} - \hat{y}_{\text{pred}}|$$

    MAE measures the average absolute difference between the predicted values ($\hat{y}_{\text{pred}}$) and the true values (y_{true}) of a regression model, where n is the number of instances. MAE is frequently used as an alternative to MSE. MAE provides a more robust measure of errors as it is not sensitive to outliers.

  - **R-squared (R2) Score**:

$$\text{R2} = 1 - \frac{\sum_{i=1}^{n}(y_{\text{true}} - \hat{y}_{\text{pred}})^2}{\sum_{i=1}^{n}(y_{\text{true}} - \bar{y}_{\text{true}})^2}$$

    R2 represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, where a higher value indicates a better fit of the regression model.

3.3.7 Fine tuning

Hyperparameters are parameters that are not learned from data. Instead, the user sets them before training the model. There are general hyperparameters that can affect the overall training process. The *learning rate* determines the step size at each iteration, influencing how quickly the model learns. The *number of epochs* defines the number of times the model iterates over the training data. *Batch size* determines the number of training examples processed together before updating the model's weights. *Learning rate decay* reduces the learning rate over time, based on given criteria. Then there are hyperparameters which are related to the architecture of the neural network. The *number of layers* determines the depth of the network, and the number of *neurons per layer* defines the width of the network, influencing its expressive power. *Dropout rate* (Section 3.2.5) defines

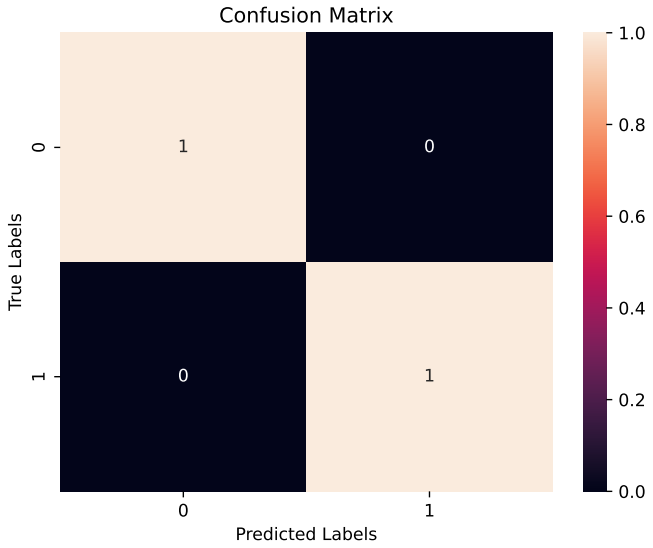


Figure 3.4: Confusion matrix with all classes perfectly predicted.

the fraction of neurons that are randomly disabled or set to zero during the training of a neural network, introducing a form of regularization to prevent overfitting (Section 3.3.5) and lower variance. The choice of an appropriate *activation function* (Section 3.2.3) is crucial as it introduces non-linearity to the network, allowing it to learn complex patterns. An optimizer (Section 3.3.3) is a key component in training a model as it iteratively adjusts the weights and biases based on the *loss function* (Section 3.3.2), which quantifies the difference between the predicted output and the actual output, ultimately aiming to minimize the loss and optimize the model's performance.

There are several techniques that help fine-tune a model and find the best hyperparameters. Grid search systematically explores a predefined grid of hyperparameter values, exhaustively evaluating each combination, making it computationally expensive and time-consuming, especially when dealing with a large number of hyperparameters or a wide range of values. Random search, on the other hand, randomly samples hyperparameter combinations within a specified search space, but there's a possibility of missing important combinations. Bayesian optimization combines the advantages of both techniques by constructing a probabilistic model of the performance, focusing the search on promising areas. It adapts its exploration based on the evaluation results, efficiently narrowing down the search space. However, it requires careful tuning of its own hyperparameters and can be more complex to implement.

3.4 Types of neural networks

Neural networks come in various types, each tailored for specific tasks and data patterns. In this section, we will explore the different types of neural networks, their characteristics, and their applications.

- **Feed-forward neural networks (FFNNs)**, also known as multi-layer perceptrons (MLPs), are the simplest and most commonly used type of artificial neural network. FFNNs are made of multiple layers of interconnected neurons that process input data and produce output, without any feedback loops, hence the name "feed-forward".

  Each neuron applies a (linear or nonlinear) activation function on input from the previous layer and produces an output that is passed on to the next layer. The neurons in the input layer

have no activation function, while those in the hidden and output layers can use different activation functions (Section 3.2.3).

FFNNs are trained using a supervised learning approach (Section 3.1), where the network learns from labeled examples of input-output pairs. During training, the network adjusts the weights and biases of its neurons to minimize the error between the predicted output and the actual output. This is achieved using backpropagation (Section 3.3.4).

- **Convolutional Neural Networks (CNNs)** are a type of deep neural network commonly used in signal processing (1D) and image (2D) applications. It was first introduced by Yann LeCun in the late 1980s. LeCun developed the first convolutional neural network, designed to recognize handwritten digits and used for processing checks in the banking industry (Lecun et al., 1998). The key difference between a CNN and a traditional neural network is convolutional layers, which apply a set of learnable filters to the input data. These filters have small spatial dimensions and slide over the input data, performing element-wise multiplication and summation operations. The result of this operation is a new feature map that highlights regions in the input data that are similar to the filter. By stacking multiple convolutional layers, the network can learn increasingly complex and abstract features, using backpropagation (Section 3.3.4) and gradient descent (Section 3.3.3).

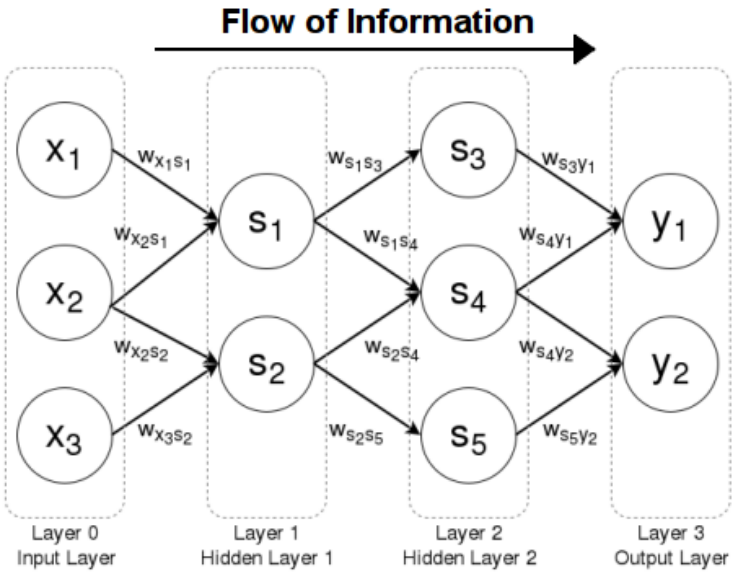


Figure 3.5: FFNN scheme[3].

After the convolutional layer, a pooling layer is often added to reduce the spatial size of the feature maps and make the network more computationally efficient. The pooling operation typically involves taking the maximum or average value over a small region of the feature map. This has the effect of downsampling the feature maps and reducing their resolution, which also helps to introduce some degree of translation invariance to the feature maps, making the network more robust to variations in the position of the features in the input.

The formula for convolution operation in a 1D convolutional layer:

$$h_j^k = \sigma\left(\sum_{u=1}^{f}\sum_{l=1}^{c} w_{ul}^k x_{(j-1)s+u,l} + b_k\right), \tag{3.2}$$

[3] Downloaded from https://brilliant.org/wiki/feedforward-neural-networks/.

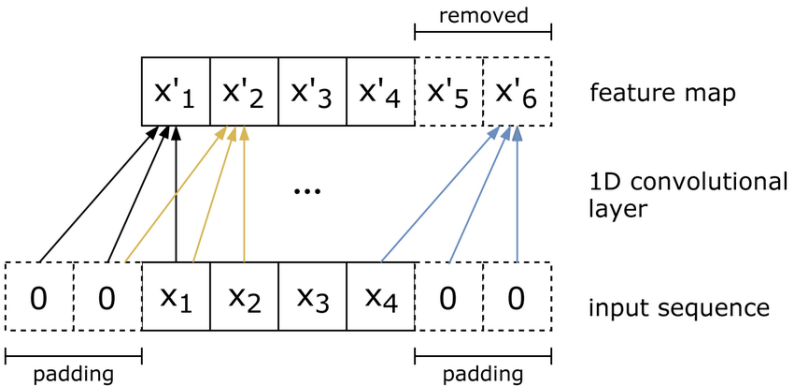


Figure 3.6: 1D CNN scheme (Castro et al., 2019).

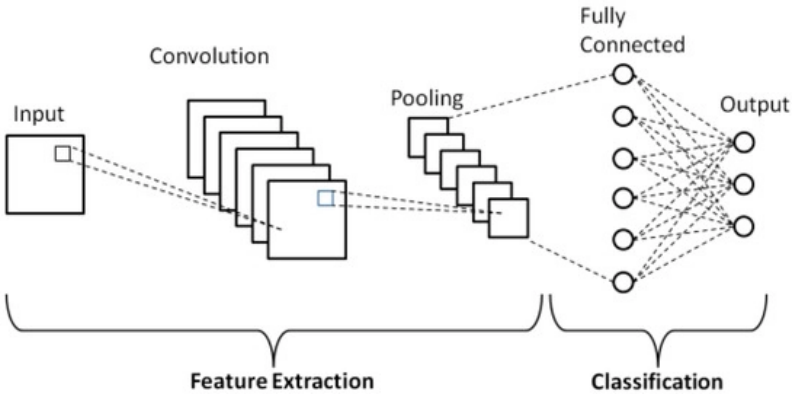


Figure 3.7: 2D CNN scheme (Phung & Rhee, 2019).

where index j represents the index of the neuron within the layer, index k represents the index of the layer, h_j^k is the output of the neuron, $\sigma(\cdot)$ represents the activation function, w_{ul}^k represents the weight of the connection between the l-th input in the previous layer and the j-th neuron in the current layer, $x_{(j-1)s+u,l}$ represents the l-th input value from the previous layer, specifically from the $(j-1)s+u$-th neuron, where s represents a stride factor, f represents the filter size and c represents the number of channels in the input.

The output of the convolutional and pooling layers is then flattened and passed through one or more fully connected layers (similar to the ones in a traditional neural network), which perform the final classification or regression.

In addition to these layers, CNNs may also include normalization (Section 3.2.4) layers, which can help prevent the network from being affected by differences in the scale of the features (different instrument, same physical origin - pattern), making the network more robust to variations and can help prevent overfitting, speed up training (it is faster with smaller numbers), and dropout layers that randomly drops out a fraction of the neurons in the previous layer to prevent overfitting.

- **Recurrent Neural Networks (RNNs)** Recurrent neural networks are designed to process sequential data, such as text or speech. They are composed of a feedback loop that allows the output of a neuron to be fed back into the network as an input. RNNs can remember past inputs and use that information to influence future outputs. One type of RNN is Long short-term memory (LSTM) which can remember longer input data sequences.

- **Long Short-Term Memory (LSTM)** is a type of recurrent neural network (RNN), which

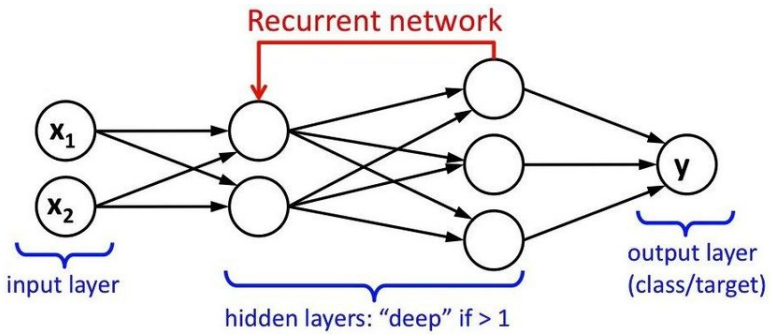


Figure 3.8: RNN scheme (Mishra et al., 2018).

was first proposed by Hochreiter and Schmidhuber in 1997 (Hochreiter & Schmidhuber, 1997), that is especially effective in processing sequential data such as handwriting, speech, language translation and most importantly time-series. Unlike traditional RNNs, which suffer from the vanishing gradient problem (Section 3.2.3) that limits their ability to learn long-term dependencies, LSTMs are designed to maintain a memory state that allows them to remember important information over extended periods of time.

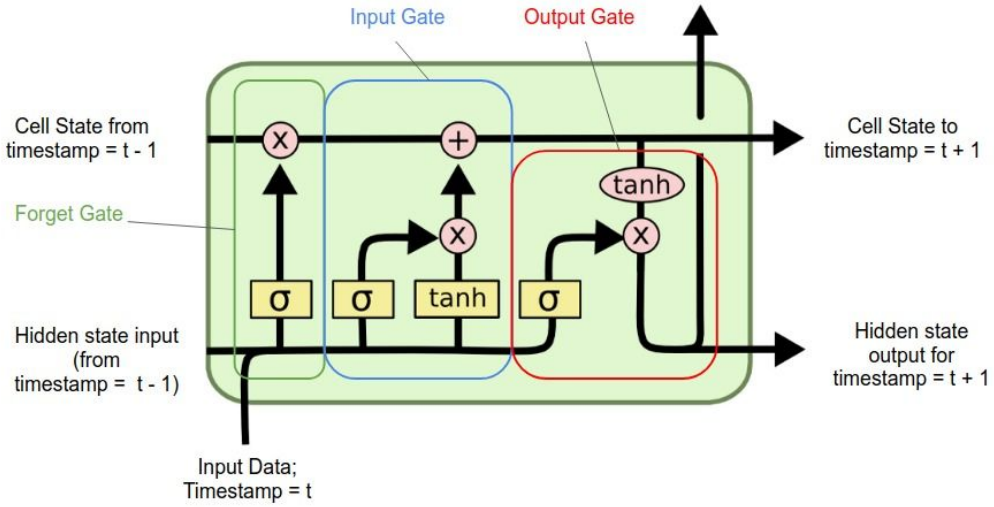


Figure 3.9: LSTM scheme[4].

The core idea behind LSTMs is to introduce a set of memory cells controlled by three types of gates: the input gate, the forget gate, and the output gate. The input gate determines how much new information should be added to the memory cells, and the forget gate decides what information should be discarded from the memory cells. The output gate controls how much information is read from the memory cells to produce the output of the LSTM. By carefully tuning these gates, LSTMs can learn to model complex sequences and capture long-term dependencies in the data.

Overall, LSTMs have proven to be a powerful and versatile tool for modelling sequential data. Their ability to handle long-term dependencies has made them a popular choice in many fields.

[4] Downloaded from https://medium.com/analytics-vidhya/lstms-explained.

- **Autoencoders**

  The architecture of an autoencoder typically consists of three main parts: an encoder, a bottleneck, and a decoder. The encoder takes the input data and maps it to a lower-dimensional representation, which is then passed through the bottleneck. The bottleneck has fewer neurons than the input and output layers for it to be able to capture the most essential features. The decoder then maps the bottleneck representation back to the original input space. Autoencoders have several advantages in unsupervised learning, including reducing dimensionality, detecting anomalies, learning meaningful features, and generating new data.

  There are many different types of autoencoders, each with unique architecture and training objectives. Some common types include (i) Denoising autoencoder: This type of autoencoder is designed to remove noise from input data. During training, the autoencoder is presented with noisy input data and is trained to generate a clean output; (ii) Variational autoencoder: This type of autoencoder is designed to learn a probability distribution over the encoded latent space rather than a deterministic mapping. This allows the autoencoder to generate new data samples by sampling from the learned distribution; (iii) Convolutional autoencoder: These neural networks are particularly useful for time-series data or image data, where the (spatial) structure of the data is essential; (iv) Recurrent autoencoder: RAEs can be used for a variety of tasks, such as sequence prediction and anomaly detection. For example, in sequence prediction, the RAE can be trained on a dataset of time series data and used to predict future values in the sequence. In anomaly detection, the RAE can be used to detect unusual patterns in the input sequence that may indicate a problem or anomaly. In sequence generation, the RAE can be used to generate new sequences by sampling from the compressed representation and decoding using the decoder.

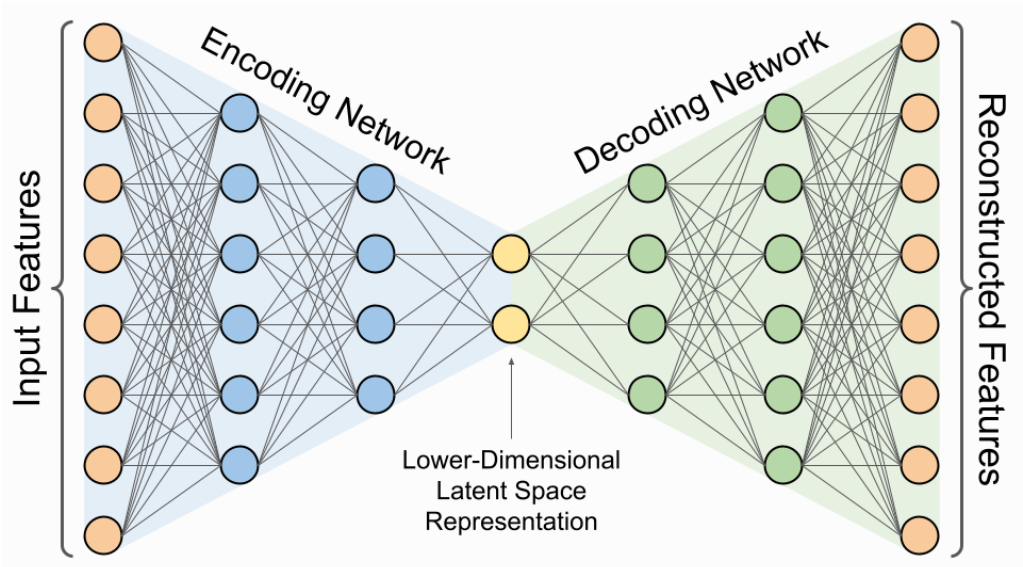


Figure 3.10: Autoencoder scheme[5].

[5] Downloaded from https://www.assemblyai.com/blog/introduction-to-variational-autoencoders-using-keras/.

Chapter 4

Methods

The main goal of this thesis is GRB detection in light curves from GRBAlpha data. Classification models based on convolutional, LSTM, and dense layers were employed to identify if the given time window contains significant gamma-ray emission or background. Three approaches were investigated; (i) classification to GRB/background (Transient detection classifier), (ii) classification based on denoised light curves by autoencoders, (iii) predicting background by LSTM regression, which could reveal excess emission.

In the first approach, the task involved classifying between Gamma-Ray Bursts (GRBs) and background signals. The input for this classification was a time window, and the output was a floating point number between 0 for the background and 1 for GRB. To tackle this problem, a combination of 1D convolutional, LSTM, and dense layers was employed.

In the second approach, an autoencoder was used for denoising the light curves, followed by the classification of the denoised data. It is known that noise is not compressible (Saliwanchik & Slosar, 2022), making it difficult to remove using traditional methods. To address this issue, two types of autoencoders were employed: one based on dense layers and another based on convolutional layers. The input and output of the autoencoder were the light curves themselves, and the model was trained to reconstruct the clean version of the light curve by minimizing the Mean Squared Error loss.

Lastly, in the third approach, an LSTM-based model was utilized for background prediction. The goal was to predict future data points in the time series based on the previous window and compare them with the real observations. By identifying any significant deviations or excesses, the model could determine the presence of a GRB. This approach leveraged the sequential nature of the data and the ability of LSTMs to capture long-term dependencies, allowing for accurate predictions and the detection of anomalies in the background signal.

4.1 Data preprocessing

The data utilized in this thesis were obtained by the GRBAlpha satellite. Most of the data points were observed using a bin size of 1 s, however, especially in the early operational phase of the satellite, also longer time bins were used. The distribution of bin sizes can be found in Table 2.2 and their visual representation in Figure 2.1. Table 2.3 serves as a reference for the format of the raw data, which needs to be further filtered and preprocessed. Only specific data columns were utilized in the analysis, namely the “start” and “end” timestamps as well as the exposure duration extracted from Table 2.3a. All relevant information was extracted from Tables 2.3b and Table 2.3d, while the temperature data from Table 2.3c was disregarded due to being captured only every minute. The exposure time and satellite position were averaged by calculating the mean of the “start” and “end” timestamps.

4.1.1 Satellite position

Spatial data given by GRBAlpha are in the Geographic Coordinate System (GCS). The drawback of this system is the wrap-around effect, meaning that locations near the International Date Line may have their longitude values abruptly change from positive (180°) to negative (-180°) or vice versa. So transformation process is required. I have chosen to convert these coordinates into XYZ (Cartesian) coordinates and scale them, for the same reason explained in the previous subsection.

4.1.2 Time conversion

The time format of YY-MM-DD hh:mm:ss is not suitable for model training, so the dates need to be converted to an integer or floating point number, namely Julian Date format was used. To avoid calculating with large numbers, Julian date of the first light was subtracted i.e. 2459319.

4.1.3 Filtering data

As said in Section 2.1, there are some data that were excluded from the analysis. Namely, data with bins 3,15,40, and 60 seconds, ADC channel 0 to 64, data with high spectral resolution (64 and 256), and six sets due to bad time synchronization. A total of 6,120 data points with zero counts, which were identified as originating from internal software problems, were also excluded from the analysis. These points are undoubtedly invalid and do not represent noise (the second lowest number of counts in the dataset observed is 27).

4.1.4 Rebinning

For the background estimating model, we used training data with 4-second bins. In order to utilize also the 1 and 2-second bin data, we decided to rebin 1s and 2s bins into 4s bins. This approach allowed for the utilization of a larger dataset while maintaining a consistent binning interval. For the remaining models (GRB classification, AE denoising), we only used data with 1-second time bins.

4.1.5 Sliding window

Processing continuous temporal data is not particularly well-suited for employed model architectures. In order to process the data, the data need to be segmented into smaller intervals of fixed size using a sliding window method. In the case of transient detection classifier, such an approach can also be utilized to produce continuous output and make the prediction more robust.

The sliding window method is commonly used when applying LSTM for background prediction. This technique involves sliding a fixed-size window over the time series data, generating multiple overlapping sequences, and increasing the amount of training data. This can improve the LSTM model's ability to generalize and make accurate predictions.

Moreover, the sliding window method can emulate real-time prediction situations by refreshing the window with every incoming data point and act as a triggering system. If the window does not match with the prediction for an interval of time, it can be considered a trigger.

4.1.6 Scaling and normalization

Transformation and normalization of the input data are important in machine learning applications because they can help to equalize the scales of different features. When features have different scales, it can lead to biased results and inefficient convergence of optimization algorithms, also avoiding numerical instability and allowing the model to focus on relative relationships rather than absolute values. For these reasons, we performed a logarithmic transformation of count rates. In

addition to logarithmic scaling, other methods, such as scaling between 0 and 1 and normalization, were also considered, but logarithmic scaling demonstrated the best overall performance. It also helps in handling missed outliers. Further normalization of the input data is performed by having the BatchNormalization layer as the first layer in every model (Section 3.2.4).

4.2 Transient detection classifier

4.2.1 Architecture

The model architecture consists of several layers: a 1D convolutional layer (Conv1D) with 64 filters, a kernel size of 5, and a ReLU activation function, capturing local patterns in the input sequence; a max pooling layer (MaxPooling1D) with a pool size of 2, downsample and retain the most salient features; a Long Short-Term Memory (LSTM) layer with 32 units, capturing long-term dependencies in the data; a flattening layer (Flatten); a dense layer (Dense) with 32 units and ReLU activation; and finally, a dense layer with 1 unit and sigmoid activation, which produce a probability score indicating the likelihood of a GRB within the input time window. Overall, this model architecture aims to leverage the strengths of convolutional and recurrent layers to capture both local and long-term patterns in the input sequence, making it suitable for tasks involving sequential data.

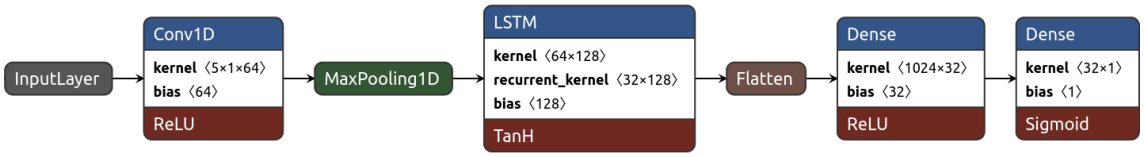


Figure 4.1: Transient detection classifier architecture.

4.2.2 Training

For this model architecture, 4 sets of training data were employed: model **SIM** trained with only simulated data (50/50 GRB/background) (Section 2.3,) model **SIM+BKG250** with simulated data and real background with the time windows having mean under 250 counts, model **SIM+BKG500** is similar but with mean under 500 counts, and last one model **SIM+BKG250+AUG** with simulated data (50/50 GRB/background), time windows means under 500 and with 11 augmented real GRBs (Section 2.3.1). For every picked real GRB, 100 augmented ones were generated. The model is compiled with the Adam optimizer, a learning rate of 0.0001, and the binary cross-entropy loss function.

Table 4.1: Number of Windows for Each Dataset and Model

Dataset	**Number of Windows**
Simulated	1,000,000
Real Background (Mean < 250)	1,223,774
Real Background (Mean < 500)	1,356,441
Augmented Peaks	52,800

4.2.3 Testing

In the task of detecting GRBs, various metrics are utilized to assess the performance of the model. The model is evaluated using both simulated and real data. Since the model outputs likelihood probability scores, it is necessary to set an optimal threshold to classify them into background and GRB categories. To determine this threshold, heatmaps were generated, plotting different SNR ratios against threshold values ranging from 0 to 1. This process was repeated for 1000 simulated GRBs and backgrounds.

For training and testing purposes, only 11 GRBs detected by GRBAlpha were selected (Section 2.3.1). Given our knowledge of the GRB locations, we extracted 64-bin time windows with the peak amplitude positioned in the middle. From these windows, we calculated the mean and standard deviation using the first 20 bins and the last 5 bins. The peak value was determined as the maximum value within the window. All real backgrounds with 1-second bins were divided into training and testing samples.

In addition to the ROC curve, AUC score, and confusion matrix, a histogram was created to display the predicted values for real GRBs and backgrounds. This histogram provides both numerical and visual insights into the classification performance.

4.3 Denoising autoencoder

4.3.1 Architecture

Two variants of the autoencoder model were studied: one utilizing dense layers and the other employing 1D convolutional layers. Both had similar structures; the encoding and decoding part had 3-5 dense/convolution layers with descending number of units/filters ranging from 64 to 16, with convolution layers having kernel sizes from 3 to 8, bottleneck with size 8 or 16 and all using ReLU activation function.

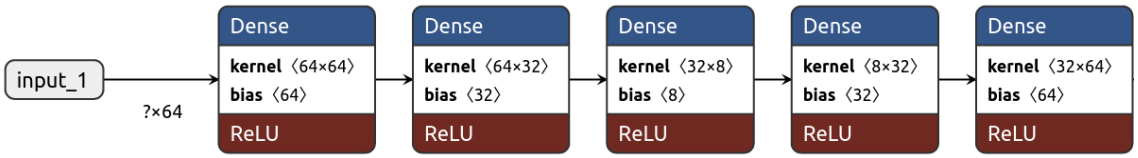


Figure 4.2: Example of used autoencoder architecture.

4.3.2 Training

Autoencoders were training using 852,512 4-second bins, with a training/testing split of 90/10. The training process involved 200 epochs, but to prevent overfitting, an EarlyStopping callback was utilized, so generally never crossed 100 epochs. Additionally, a ReduceLROnPlateau callback was employed, reducing the learning rate by a factor of 0.3 if the validation loss did not improve for 4 consecutive epochs. Several loss functions were examined, including mean square error (MSE) and mean absolute error (MAE). Subsequent experimentation revealed that the utilization of MAE resulted in superior performance compared to MSE.

4.3.3 Testing

Several metrics were used to assess the performance of the model. These metrics included mean squared error (MSE); mean absolute error (MAE); root mean squared error (RMSE). By utilizing a combination of these metrics, the effectiveness of the denoising autoencoder in removing noise while preserving important features were thoroughly evaluated.

4.4 Background predicton

4.4.1 Architecture

The LSTM model was tried due to its ability to capture long-term dependencies effectively. The model consists of multiple LSTM cells, each containing an input gate, forget gate, and output gate. These gates regulate the flow of information within the cell, allowing it to retain and update information. In our implementation, input data were time windows (Section 4.1.5), 1-3 LSTM layers with 16-128 hidden units in each layer. Following the LSTM layers, non or up to 3 dense layers with 4-128 units, to further process the LSTM outputs and capture higher-level representations. This combination of LSTM and dense layers enables the model to learn complex patterns and relationships in the data.

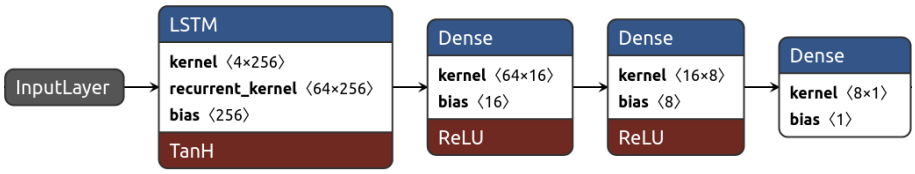


Figure 4.3: Example of used LSTM architecture.

4.4.2 Training

LSTM model was trained on 852512 data points of 4s bins (90/10 split for training/testing). The number of epochs was set to 200, but EarlyStopping callback was employed, and so the models almost never crossed 100 epochs. Also, ReduceLROnPlateau callback was added, with a factor of 0.5, with the condition of validation loss not improving for 4 epochs. Different loss functions were tested, including mean square error, mean absolute error, Huber loss, or even custom ones.

4.4.3 Testing

For the LSTM background prediction model, several metrics were employed to evaluate its performance. Firstly, the mean absolute error (MAE). This metric provides a measure of the model's accuracy in capturing the magnitude of the background fluctuations. Additionally, the root mean square error (RMSE) was utilized, considering both magnitude and direction. The coefficient of determination (R-squared) was employed to determine the proportion of variance in the background data. This metric indicates how well the LSTM model captures the underlying patterns and trends in the background data.

4.5 Implementation

For the implementation of below stated neural networks, I have chosen Python due to its simplicity, flexibility, and powerful libraries, e.g., *Pandas* (Reback et al., 2020) and *Numpy* (Harris et al., 2020) for preprocessing and data-handling, *Scikit-learn* (Pedregosa et al., 2011), *TensorFlow* (Abadi et al., 2015) with API *Keras* (Chollet et al., 2015) for building machine learning models.

As said before, the use of graphics cards (GPUs) is essential in deep learning because they can significantly speed up the training process of large neural networks. As computing power to carry out the experiments efficiently, I used a school server called Cthulhu, equipped with Nvidia GeForce RTX 2080 Super GPU with 8 GB of GDDR6 memory. With this computing power, the longest training duration did not exceed ten hours.

4.5.1 Callbacks

Callbacks are functions or objects that enable the execution of custom code at specific stages during the training of a model. The following callbacks are commonly employed in various models: (i) ReduceLROnPlateau reduces the learning rate when the validation loss ceases to improve (either it starts increasing or remains constant); (ii) EarlyStopping terminates the training process when the validation loss fails to improve for a specified number of epochs; (iii) ModelCheckpoint saves the entire model during training when the validation loss reaches its lowest value.

Chapter 5

Results

5.1 Transient detection classifier

For the purpose of the transient detection classifier, we employed four models that shared the same architecture (Section 4.2.1) but were trained on different combinations of datasets (Table 4.1).

Given the comparable performance of all four models on the simulated data, Figure 5.1 serves as a visual representation rather than an accurate assessment of their capabilities, with the left plot depicting an almost perfect ROC curve, the center plot showcasing a histogram of predicted values, and the right plot illustrating a confusion matrix with values expressed as percentages.

When analyzing the models' performance on simulated data, we concluded that the sufficient threshold for detections is 0.8, which maximizes the true positive rate while maintaining the false positive rate at an acceptable level for testing data (5.1). Our set threshold and for SNR above 3 . We can see in Figure 5.2 that our set threshold yields good performance for simulated GRBs above SNR $\gtrsim 3$.

Figures 5.3 show four plots, each containing predictions of real backgrounds and known GRBs from GRBAlpha by each model. SNR for the background was estimated from the bin with the highest count rate in the window.

Figure 5.4 displays a comparison of the four models by ROC curve, histogram of predicted values, and their corresponding confusion matrix. Models performance is summarized in Table 5.1.

Model	AUC
SIM	0.808
SIM+BKG250	0.455
SIM+BKG500	0.636
SIM+BKG500+AUG	1.000

Table 5.1: Models performance.

Figure 5.5 shows the prediction of model **SIM+BKG500+AUG** on windows with real GRBs. As the window slides through the data, continues region of predictions above set threshold (< 0.8).

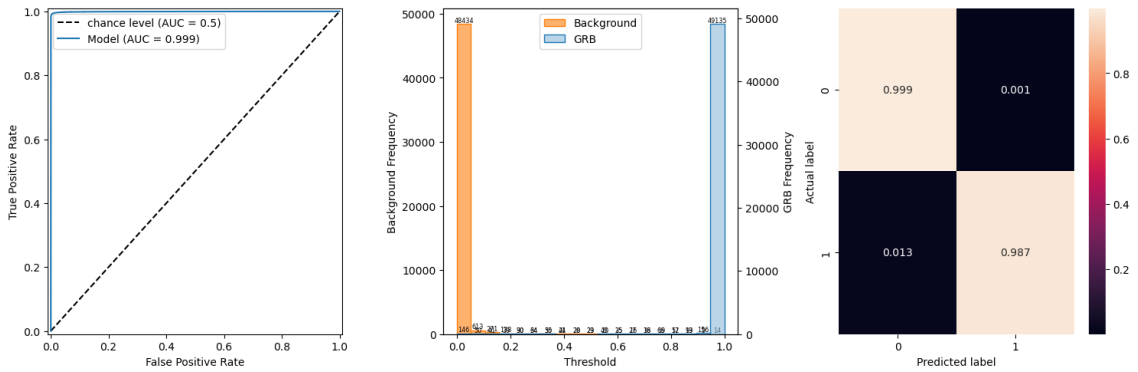
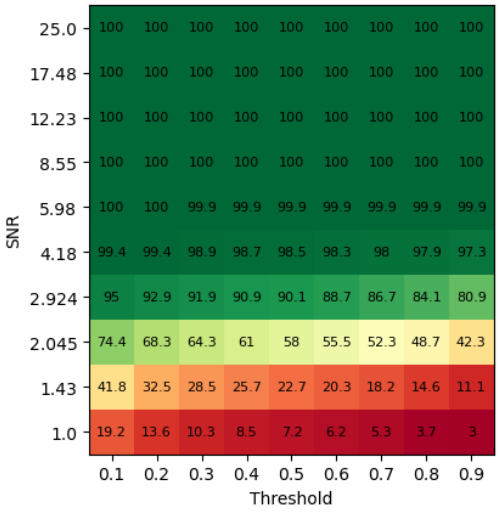
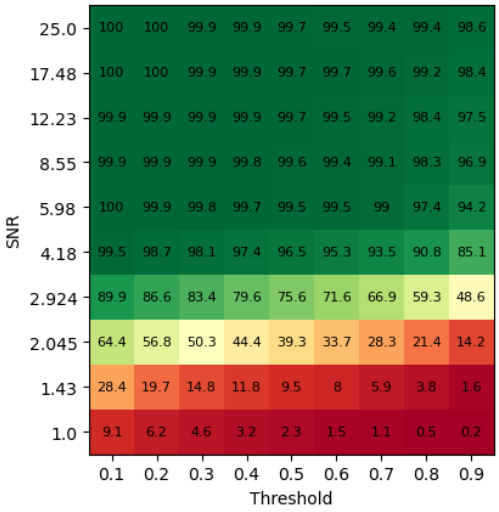


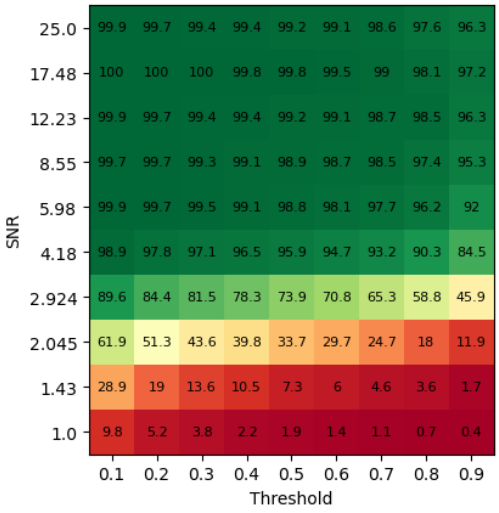
Figure 5.1: Representation of the performance of all four models on a million simulated light curves; **Left:** ROC curve, demonstrating near-perfect scores; **Center:** histogram of predicted values, showcasing the distribution of model outputs; **Right:** confusion matrix with values expressed in percentages, providing an overview of the classification accuracy for each predicted class.



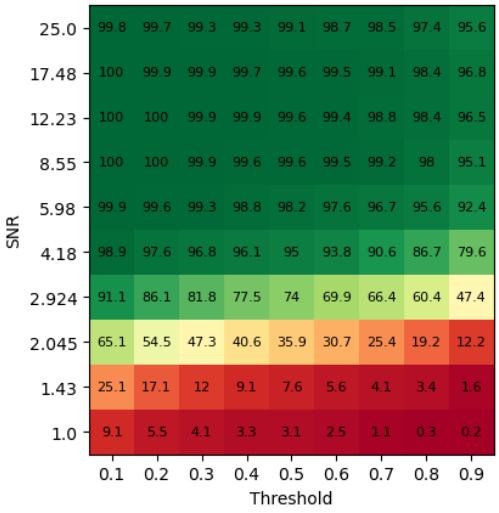
(a) model SIM



(b) model SIM+BKG250

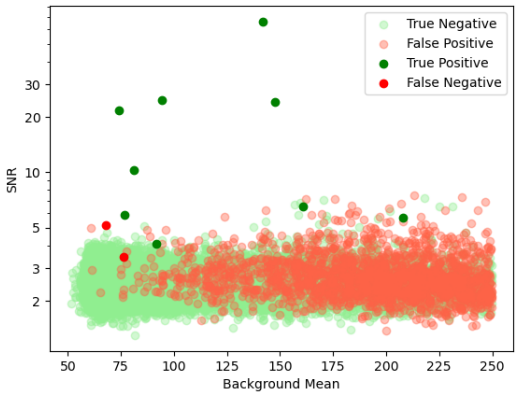


(c) model SIM+BKG500

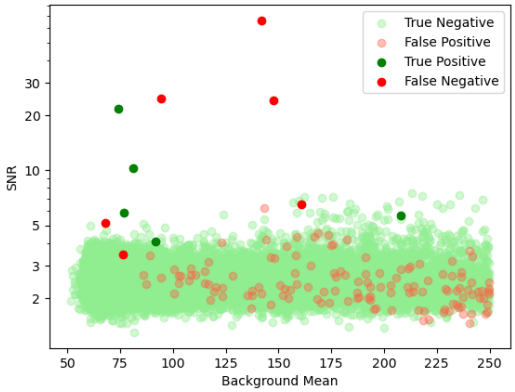


(d) model SIM+BKG500+AUG

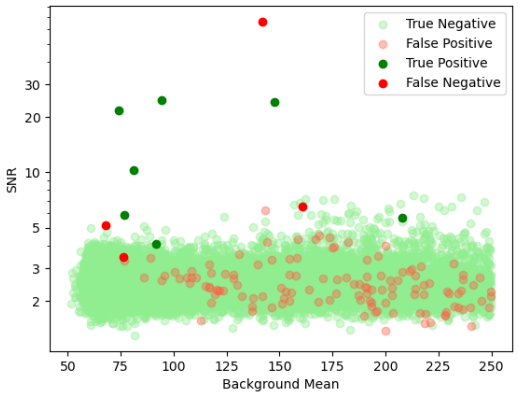
Figure 5.2: Visualization of model performance on GRB data based on SNR. The plot illustrates the number of predictions above different threshold values for various SNR levels. Each cell represents the percentage of predictions above the threshold, with SNR values plotted on the y-axis and threshold values on the x-axis. The color intensity indicates the proportion of predictions above the threshold, with darker shades indicating a higher percentage.



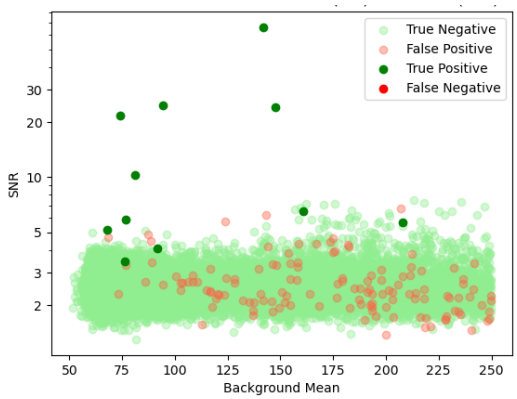
(a) model SIM



(b) model SIM+BKG250

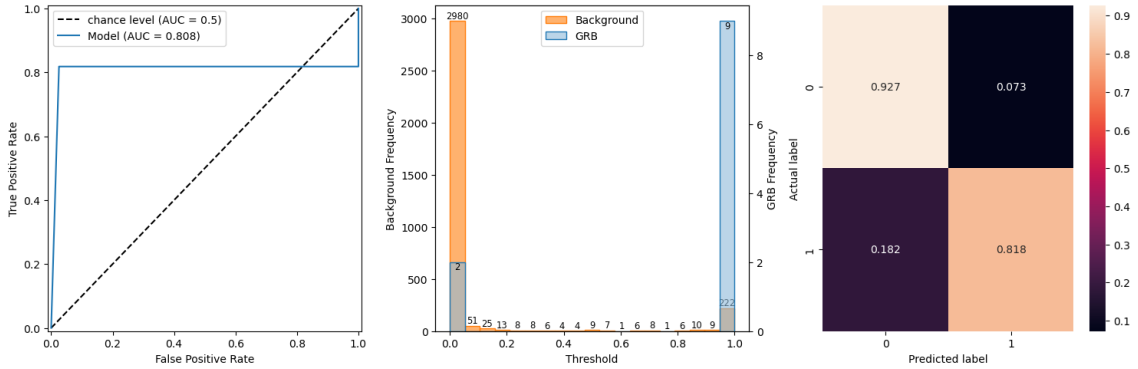


(c) model SIM+BKG500

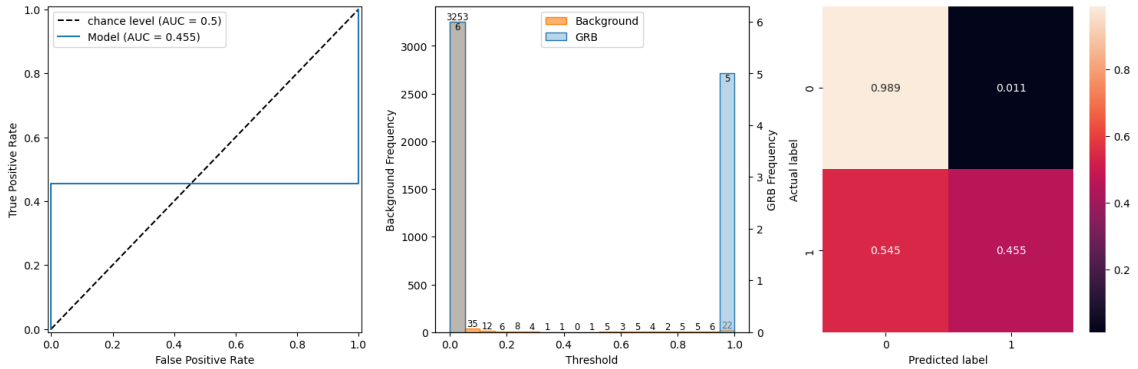


(d) model SIM+BKG250+AUG

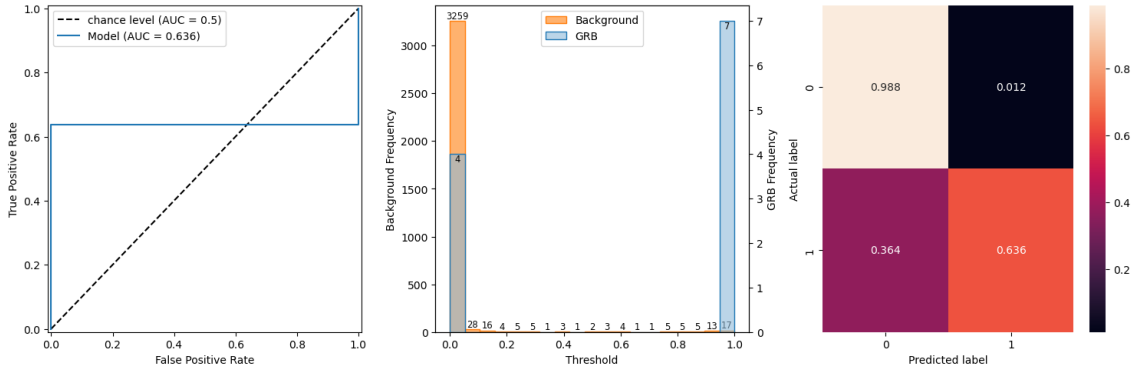
Figure 5.3: Scatter plot depicting the performance evaluation of models on 3378 real background windows and selected 11 GRBs, illustrating the relationship between the background mean of the window and SNR. True negatives are represented by light green markers, false positives by tomato-colored markers, true positives by green markers, and false negatives by red markers. The plot showcases the distribution of predicted outcomes based on the model's threshold (< 0.8), with the x-axis representing the background mean and the y-axis representing the SNR. The logarithmic scale is used for the y-axis.



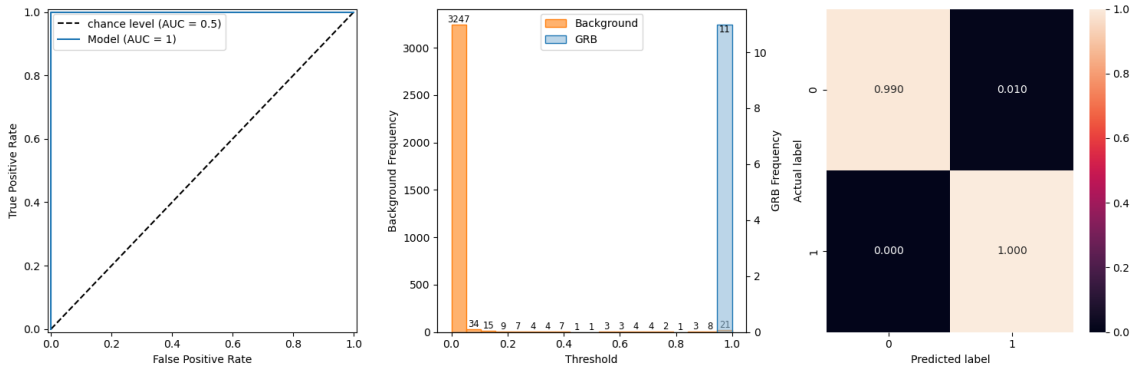
(a) model SIM



(b) model SIM+BKG250



(c) model SIM+BKG500



(d) model **SIM+BKG500+AUG**

Figure 5.4: Accuracy metrics for all four models on real GRB light curves; **Left:** ROC curve; **Center:** histogram of predicted values, showcasing the distribution of model outputs; **Right:** confusion matrix with values expressed in percentages, providing an overview of the classification accuracy for each predicted class.

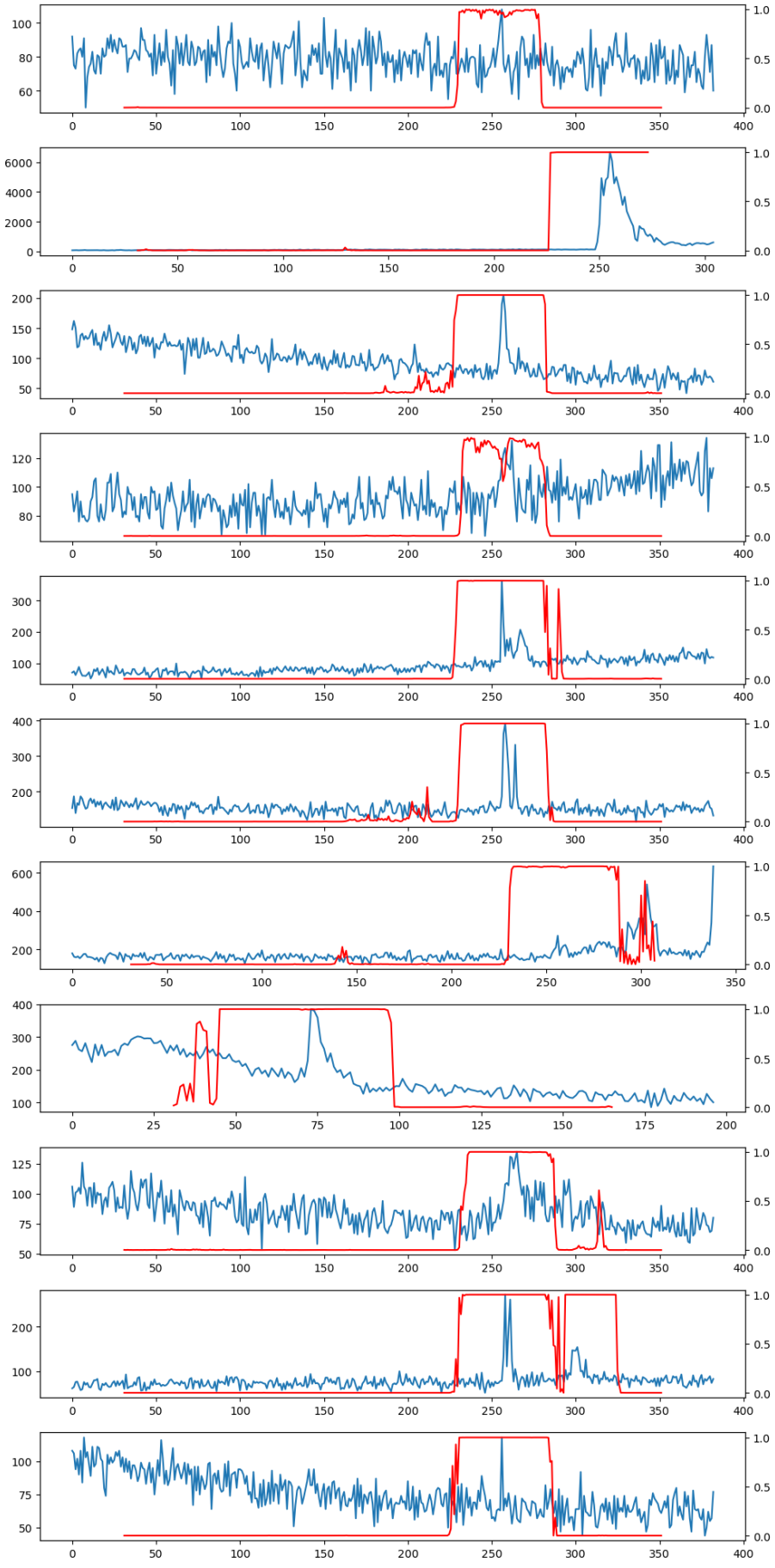


Figure 5.5: Prediction of model **SIM+BKG500+AUG** on windows with real GRBs.

5.2 Denoising autoencoder

After conducting thorough experimentation and evaluation, I have made the final decision not to employ autoencoders due to their disappointing performance. Despite our initial optimism and significant investment in training the models, they consistently demonstrated limitations and fell short of meeting our expectations.

Figure 5.6 shows examples of the poor performance of these models on 4 GRBs and background in 4-second bins.

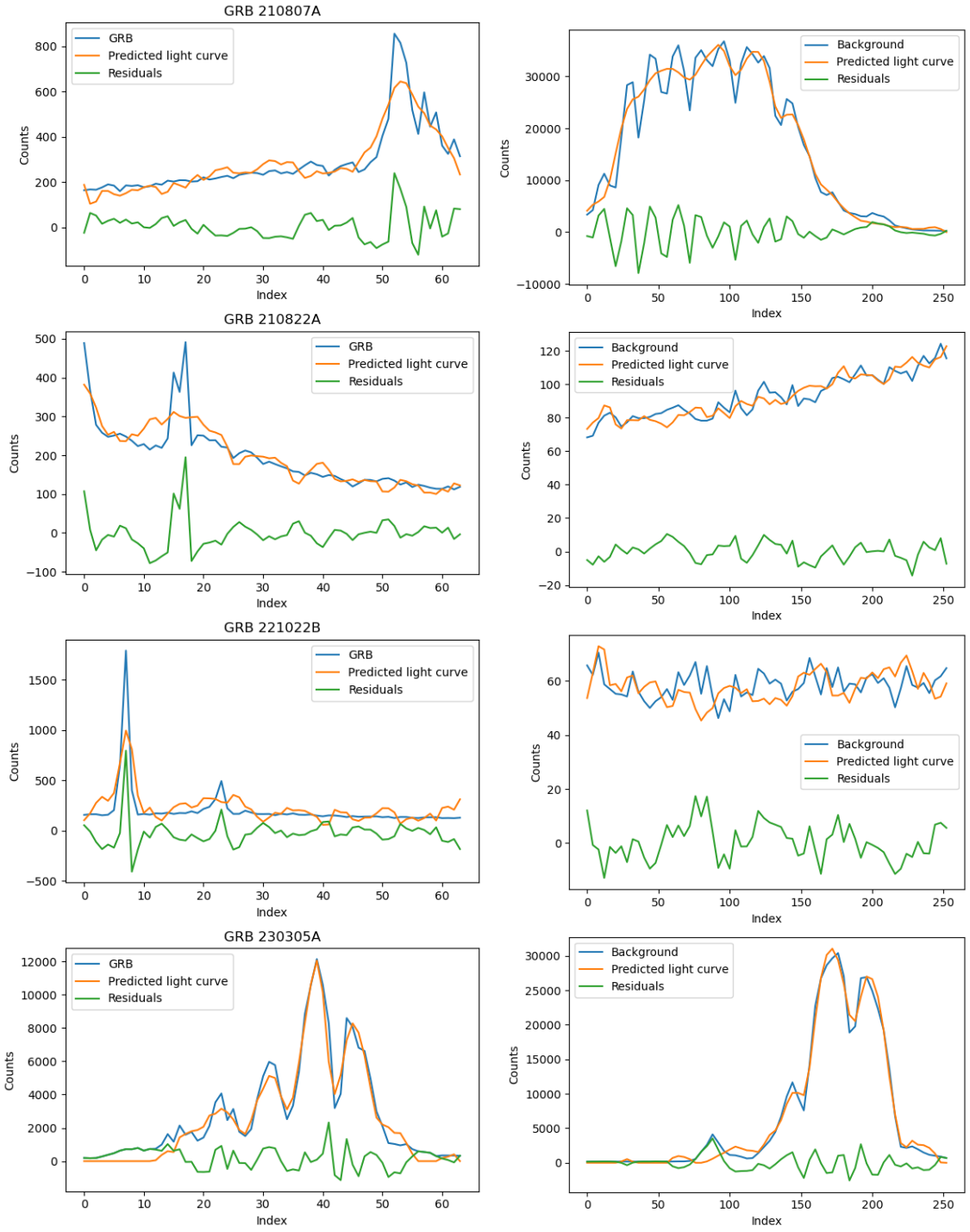


Figure 5.6: Autoencoder application on **Left:** selected GRBs, **Right:** various background

5.3 Background prediction

Unfortunately, despite our initial hopes and extensive experimentation and evaluation, we ultimately decided against utilizing trained models for background prediction due to their unsatisfactory performance. Despite initial optimism and investment in training the models, they consistently exhibited limitations and failed to meet our expectations. Consequently, we shifted our focus toward detection itself.

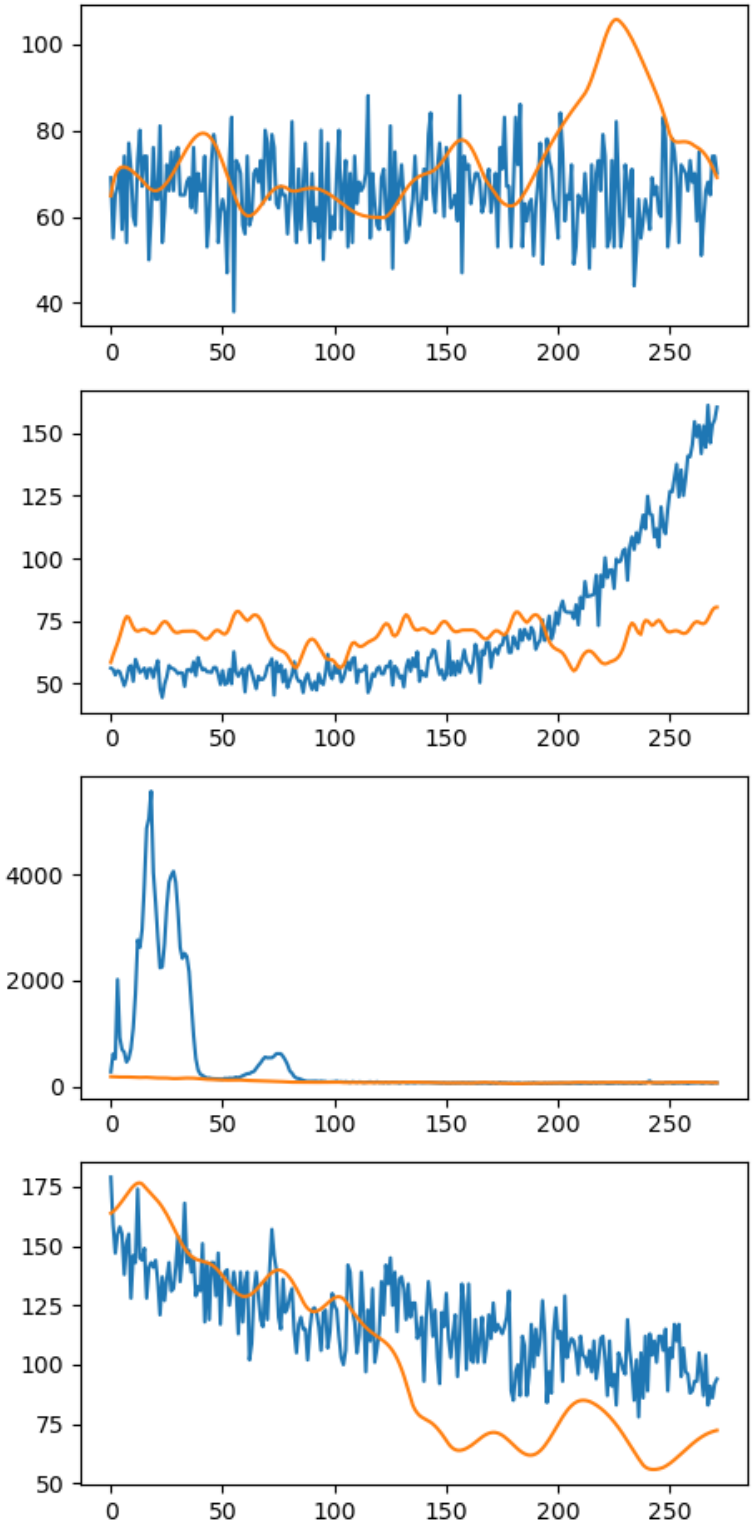


Figure 5.7: Plots showcasing the poor performance of LSTM-based background estimation.

Chapter 6

Discussion

The thesis presented three approaches for the detection of GRBs in light curves from GRBAlpha data, using various types of deep learning models. These approaches included the transient detection classifier, denoising autoencoders for pre-processing light curves, and an LSTM-based model for predicting background emission.

In terms of data preprocessing, there were numerous challenges addressed, such as bin size variability, converting geographical coordinates, and dealing with outliers. The logarithmic transformation used for count rates proved to be an effective method for normalization, which not only equalized the scales of different features but also helped in handling outliers.

6.1 Transient detection classifier

The transient detection classifier proved to be an effective tool for GRB detection. The model exhibited near-perfect performance on simulated data, with a well-defined ROC curve, suggesting that it can distinguish between background and GRB signals reliably. However, the model's performance on real data remains a critical point for further investigation.

Based on the figures presented in Section 5, a comprehensive comparison of the four models used in our study reveals distinct variations in their performance. Model SIM consistently demonstrated the poorest performance when considering the entire dataset, exhibiting limited accuracy and sensitivity in transient detection due to its tendency to generate a significant number of false positives in the background. However, it is important to note that model SIM showed relatively good performance on known GRBs, except for two cases with low SNR and low mean count 5.3a. On the other hand, models SIM+BKG250 and SIM+BKG500 exhibited lower numbers of false positives compared to model SIM, suggesting a better ability to discriminate against background events. However, this improvement came at the cost of lower numbers of true positives. Models SIM+BKG250 and SIM+BKG500 displayed reduced sensitivity in detecting transient events, potentially indicating a more conservative approach in classifying events as transients. Despite the lower number of false positives, the trade-off with a decreased number of true positives raises concerns about the overall performance of models SIM+BKG250 and SIM+BKG500 in capturing the full range of transient events. In contrast, model SIM+BKG500+AUG consistently outperformed the other models, showcasing the highest accuracy and sensitivity in detecting both known GRBs and background events. This superior performance can be attributed to its incorporation of real background and augmented peaks.

It is important to highlight that only model SIM+BKG500+AUG consistently outperformed the thresholding method in transient detection. While the rest of the models exhibited their own strengths and weaknesses, they did not consistently outperform the thresholding method in terms of sensitivity and specificity. The thresholding method relies on setting a fixed threshold (in our case, SNR 4) for event detection. These findings emphasize the superiority of the models over the traditional thresholding method and underscore their potential as valuable tools for reliable and accurate transient detection in real-world scenarios.

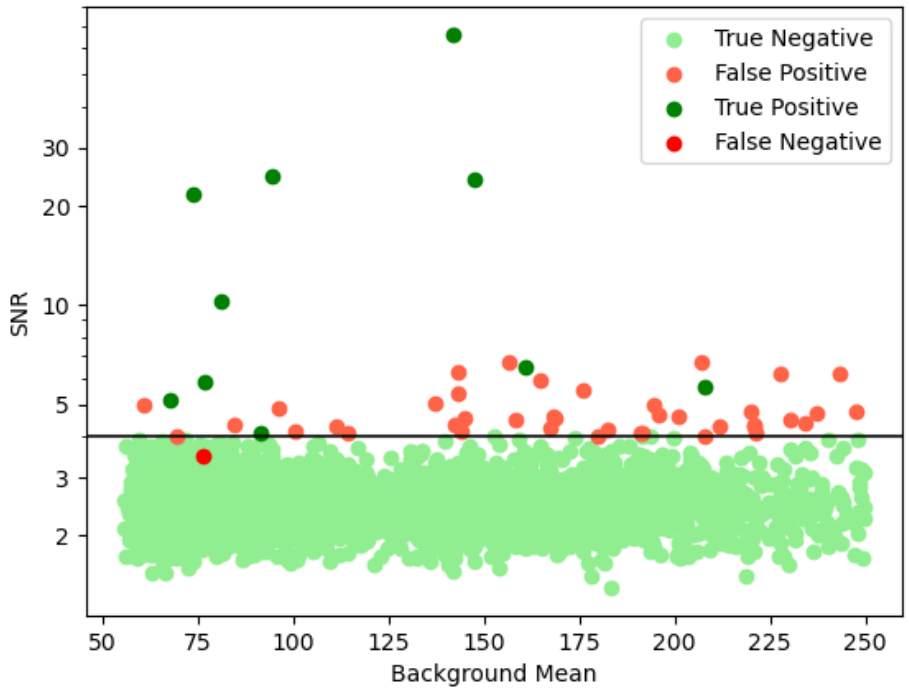


Figure 6.1: Scatter plot depicting the performance evaluation of thresholding method on 3378 real background windows and selected 11 GRBs. True negatives are represented by light green markers, false positives by tomato-colored markers, true positives by green markers, and false negatives by red markers. The plot showcases the distribution of predicted outcomes based on the model's threshold (< 0.8), with the x-axis representing the background mean and the y-axis representing the SNR. The logarithmic scale is used for the y-axis.

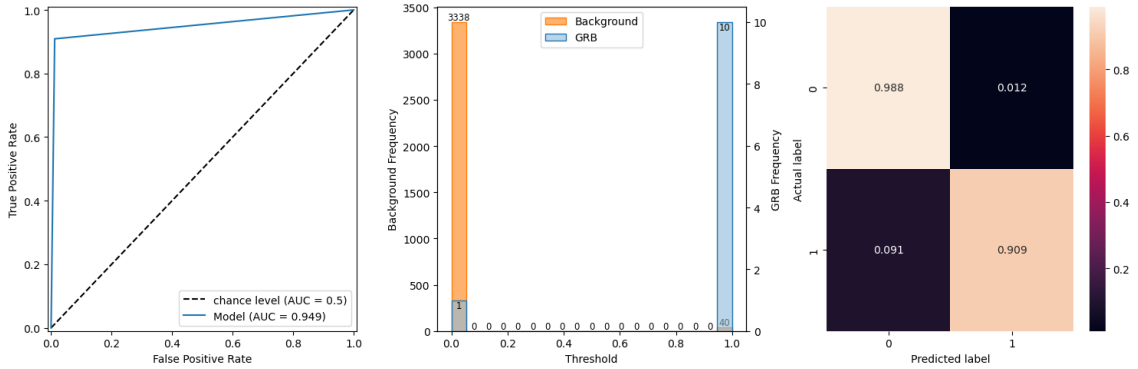


Figure 6.2: Performance of thresholding method on a real GRB and background light curves; **Left:** ROC curve; **Center:** histogram of predicted values, showcasing the distribution of model outputs; **Right:** confusion matrix with values expressed in percentages, providing an overview of the classification accuracy for each predicted class.

6.2 Denoising autoencoder

Denoising autoencoders were not utilized due to their insufficient performance in capturing the underlying patterns of the GRBAlpha background, which is not stationary. This could be for many reasons. If the noise patterns in the data were too complex or intricate due to some unpredictable motions of the satellite (in some light curves, there are prominent sinusoid variations, which suggests rotation of the satellite). Limitations in the chosen autoencoder architecture could have also hindered its performance, suggesting that alternative architectures or more advanced techniques may have been better suited for the task.

6.3 Background prediction

The main disadvantage of utilizing LSTM background prediction methods is the accumulation of errors, which occurs bin by bin. This error accumulation stems from the sequential nature of the LSTM models, where predictions are made based on previously predicted values. As the prediction process continues, any inaccuracies or deviations from the actual background values in the initial bins propagate and accumulate throughout the prediction sequence. This effect is also amplified by the fact that the LSTM layer will naturally give the most weight to the last few data points.

Due to this error accumulation, the accuracy and reliability of the LSTM background predictions may diminish over time. This is especially problematic when dealing with long-duration light curves or complex temporal patterns.

Similar problems addressed as in the previous subsection on the autoencoders contributed to the unsuccessful background estimation. Therefore I decided to drop this approach.

Conclusion

Traditionally, satellite-based systems rely on predefined thresholds or simplistic algorithms to identify these cosmic events. However, the proposed model, with its machine learning methods, has the capacity to enhance the sensitivity and accuracy of such detection systems. The model we propose for GRB transient detection showcases not only impressive capabilities (100% accuracy and O.01% false positives on real data) but also holds significant potential to be integrated as a triggering system.

Once the model is properly trained, which can take hours, the detection is fast enough to integrate as a real-time triggering system of satellites. This would allow more quick localization of afterglows, which would improve our understanding of the underlying physics behind GRBs. Specifically, future versions of the model can be tailored to suit the computing power limitations typically found in satellites.

While our proposed model holds great promise, it is important to acknowledge its current limitation with respect to background noise levels. The model, as it stands, is primarily suited for scenarios with relatively low background noise (background count less than 500 2.5). In instances where the SNR is high (Figure 5.5), the model may erroneously interpret it as a high background region, potentially leading to false negatives or reduced performance. However, it is crucial to note that this limitation could be overcome with ongoing research and development; future iterations of the model may offer improved performance in high SNR environments.

To address the issue of high background regions, several potential avenues for improvement can be explored. One approach could involve enhancing the model's training data by incorporating a broader range of GRB signatures and corresponding background noise variations or fine-tuning the model's architecture. The current approach employed in our study for transient detection involved summing the counts across all energy bands. However, upon further reflection, it is apparent that a more effective method may be to utilize multi-band light curves. This approach would allow for a more nuanced assessment, taking into account the unique spectral features that distinguish GRB emission from background noise.

Furthermore, our model can be optimized specifically for upcoming missions, for example, GRBBeta, which will be an advanced version of GRBAlpha CubeSat, which is under consideration. By tailoring the model to the characteristics of the satellite, with its sensitivity to energy bans, and using temporal and spectral data, we can achieve even better performance in detecting GRB transients, this detection method is not limited to GRB. It can be used for detecting other transients if trained on a representative sample of the transient class. This method could also be used for offline archival data transient search in already existing missions when adjusted to the satellite. Moreover, the improved performance of the optimized model extends to its integration within the CAMELOT constellation, which would benefit from a multi-satellite configuration, providing enhanced coverage and a broader field of view. By deploying the optimized model across the satellites in the constellation, the detection system can leverage collective data from multiple vantage points, resulting in a lower chance for false positives and improved overall performance.

When discussing the generation data and usage of multi-band data for our research, one

potential approach is to utilize existing sophisticated codes, such as CosmoGRB. These well-established codes have been developed specifically for modeling the emission properties of GRBs and can generate synthetic spectra that closely resemble real observations. By considering the option of using codes like CosmoGRB, we open up the possibility of incorporating more advanced and accurate simulations into our study.

Appendix

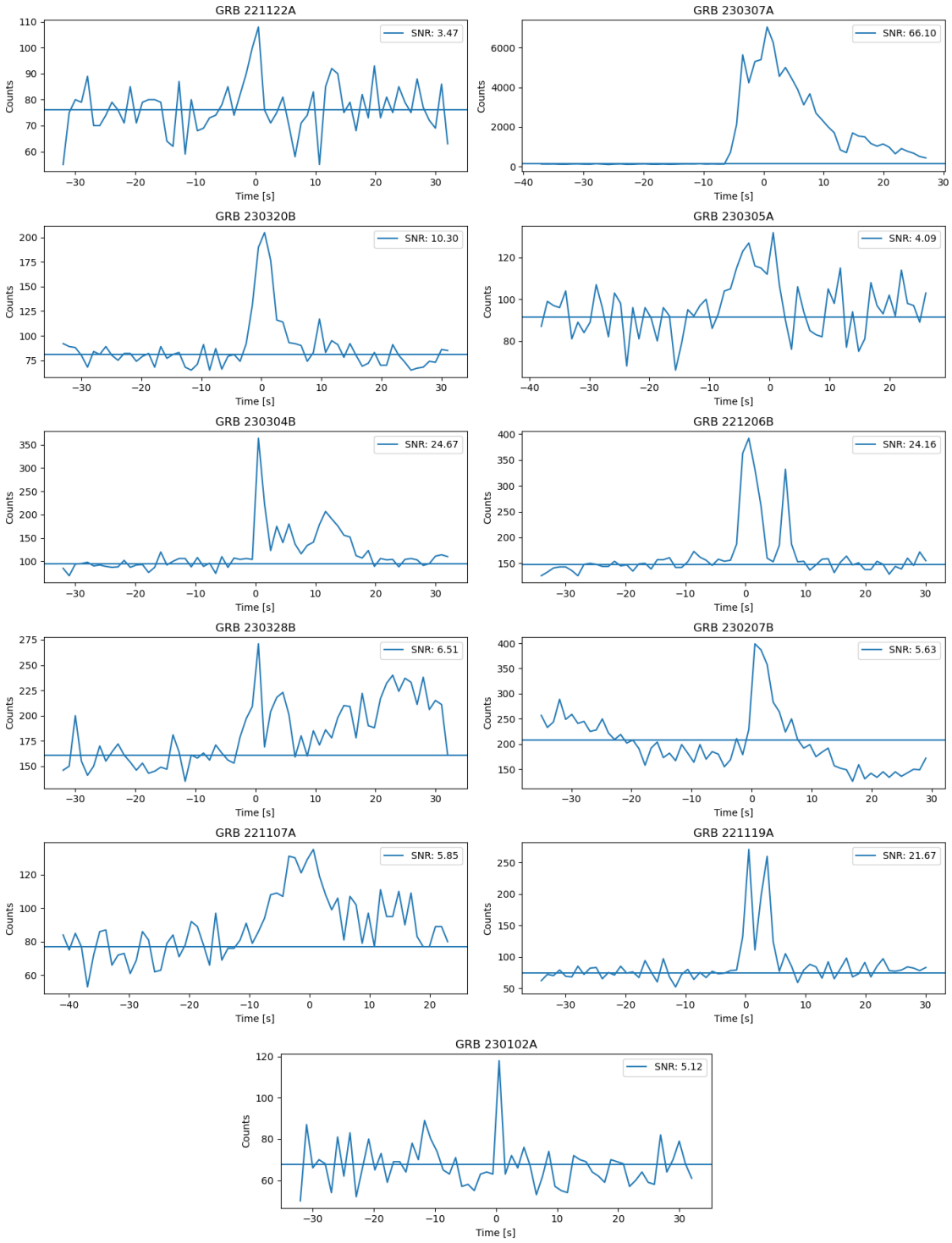


Figure 1: GRB peaks used for augmentation and testing.

Bibliography

Abadi M., et al., 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, https://www.tensorflow.org/

Abbott B. P., et al. 2017, The Astrophysical Journal Letters, 848, L12

Abraham S., et al., 2021, Monthly Notices of the Royal Astronomical Society, 504, 3084

Agarap A. F., 2019, Deep Learning using Rectified Linear Units (ReLU) (arXiv:1803.08375)

Ajello M., et al., 2021, The Astrophysical Journal Supplement Series, 256, 12

Anireh V. I., Osegi E. N., 2016, A Modified Activation Function with Improved Run-Times For Neural Networks (arXiv:1607.01691)

Atwood W. B., et al., 2009, ApJ, 697, 1071

Baker D. N., Erickson P. J., Fennell J. F., Foster J. C., Jaynes A. N., Verronen P. T., 2017, Space Science Reviews, 214, 17

Ball N. M., Brunner R. J., 2010, International Journal of Modern Physics D, 19, 1049

Band D., et al., 1993, ApJ, 413, 281

Baron D., 2019, Machine Learning in Astronomy: a practical overview (arXiv:1904.07248)

Barthelmy S., 2003, doi:10.1063/1.1361631, 35, 765

Bassi S., Sharma K., Gomekar A., 2021, Frontiers in Astronomy and Space Sciences, 8

Berger E., 2014, Annual Review of A&A, 52, 43

Bloom J. S., et al., 1999, Nature, 401, 453

Boella, G. Butler, R. C. Perola, G. C. Piro, L. Scarsi, L. Bleeker, J. A.M. 1997, Astron. Astrophys. Suppl. Ser., 122, 299

Bryson Jr A. E., Ho Y.-C., 1969, Applied Optimal Control: Optimization, Estimation, and Control. Hemisphere Publishing Corporation

Castro R., M.Souto Y., Ogasawara E., Porto F., Bezerra E., 2019, Neurocomputing, 426

Chollet F., et al., 2015, Keras, https://keras.io

Dai Z., Daigne F., Mészáros P., 2017, Space Science Reviews, 212, 409

Dubey A. K., Jain V., 2019, Lecture Notes in Electrical Engineering

Eichler D., Livio M., Piran T., Schramm D. N., 1989, Nature, 340, 126

Fishman G. J., Meegan C. A., Parnell T. A., Wilson R. B., 1982, AIP Conference Proceedings, 77, 443

Foley R. J., Kokubo M., Filippenko A. V., 2007, The Astronomical Journal, 133, 734

Fynbo J. P. U., et al., 2006, Nature, 444, 1047

Gehrels N., 2004, in AIP Conference Proceedings. AIP, doi:10.1063/1.1810924, https://arxiv.org/abs/astro-ph/0405233

Gehrels N., et al., 2006, Nature, 444, 1044

Gendre B., et al., 2013, The Astrophysical Journal, 766, 30

Getz G., Shental N., Domany E., 2006, Semi-Supervised Learning – A Statistical Physics Approach (arXiv:cs/0604011)

Ghirlanda G., Ghisellini G., Lazzati D., Firmani C., 2004, The Astrophysical Journal, 613, L13

Gokcesu K., Gokcesu H., 2021, Generalized Huber Loss for Robust Learning and its Efficient Minimization for a Robust Statistics (arXiv:2108.12627)

Goldstein A., et al., 2017, The Astrophysical Journal Letters, 848, L14

Gomboc A., 2012, Contemporary Physics, 53, 339

Gordon-Rodriguez E., Loaiza-Ganem G., Pleiss G., Cunningham J. P., 2020, Uses and Abuses of the Cross-Entropy Loss: Case Studies in Modern Deep Learning (arXiv:2011.05231)

Graves A., 2013, Generating Sequences With Recurrent Neural Networks, doi:10.48550/ARXIV.1308.0850, https://arxiv.org/abs/1308.0850

Greiner J., et al., 2010, Astronomy & Astrophysics, 526, A30

Harris C. R., et al., 2020, NumPy: Array Computing in Python, https://numpy.org/

Hochreiter S., Schmidhuber J., 1997, Neural computation, 9, 1735

Hughes J. M., 2004, Fundamentals of Space Physics. http://pages.erau.edu/~reynodb2/ep410/Hughes_FundamentalsOfSpacePhysics.pdf

Ioffe S., Szegedy C., 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (arXiv:1502.03167)

James G., Witten D., Hastie T., Tibshirani R., 2013, An Introduction to Statistical Learning: with Applications in R, 1 edn. Springer Texts in Statistics, Springer New York, NY, doi:10.1007/978-1-4614-7138-7

Kazanas D., Titarchuk L. G., Hua X.-M., 1998, The Astrophysical Journal, 493, 708

Kingma D. P., Ba J., 2014, Adam: A Method for Stochastic Optimization, doi:10.48550/ARXIV.1412.6980, https://arxiv.org/abs/1412.6980

Klebesadel R. W., Strong I. B., Olson R. A., 1973, ApJL, 182, L85

Knoll G., 2010, Radiation Detection and Measurement. Wiley, https://books.google.cz/books?id=4vTJ7UDel5IC

Kolluri J., Kotte V. K., PhridviRaj M. S. B., Razia S., 2020, 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), pp 934–938

Koskinen H. E. J., Kilpua E. K. J., 2021, Physics of Earth's Radiation Belts: Theory and Observations, 1 edn. Astronomy and Astrophysics Library, Springer Cham, doi:10.1007/978-3-030-82167-8, https://doi.org/10.1007/978-3-030-82167-8

Kouveliotou C., Meegan C. A., Fishman G. J., Bhat N. P., Briggs M. S., Koshut T. M., Paciesas W. S., Pendleton G. N., 1993, ApJL, 413, L101

Kumar P., Zhang B., 2015, Physical Reports, 561, 1

Lamb D. Q., Reichart D. E., 2000, The Astrophysical Journal, 536, 1

Lecun Y., Bottou L., Bengio Y., Haffner P., 1998, Proceedings of the IEEE, 86, 2278

Lee T., Petrosian V., 1996, The Astrophysical Journal, 474

Leon-Anaya L., Cuevas-Tello J. C., Valenzuela O., Puente C. A., Soubervielle-Montalvo C., 2023, Monthly Notices of the Royal Astronomical Society, 522, 1323

Li P., et al., 2022, Expert Systems with Applications, 205, 117296

Liu Q., Wu Y., 2012, doi:10.1007/978-1-4419-1428-6_451

MacFadyen A. I., Woosley S. E., 1999, The Astrophysical Journal, 524, 262

Mahabal A., Sheth K., Gieseke F., Pai A., Djorgovski S. G., Drake A. J., Graham M. J., 2017, in 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, doi:10.1109/ssci.2017.8280984

Martin-Carrillo A., et al., 2014, Astronomy and Astrophysics - A&A, 567, A84

Mazilu S., Iria J. P., 2011, 2011 10th International Conference on Machine Learning and Applications and Workshops, 1, 166

McCauliff S. D., et al., 2015, The Astrophysical Journal, 806, 240

McCulloh W. S., Pitts W., 1943, The bulletin of mathematical biophysics, 5, 115–133

Mészáros P., Rees M. J., 1997, ApJ, 476, 232

Mishra V., AGARWAL S., PURI N., 2018, INTERNATIONAL JOURNAL OF COMPUTER APPLICATION, 2

Mohan M., 2017.

Nair V., Hinton G. E., 2010, in Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp 807–814

Narayan R., Paczynski B., Piran T., 1992, ApJL, 395, L83

Nemiroff R. J., et al., 1993, ApJ, 414, 36

Nguyen Q., Ly H.-B., Lanh H., Al-Ansari N., Lê H., Van Quan T., Prakash I., Pham B., 2021, Mathematical Problems in Engineering, 2021

Norris J. P., 2002, ApJ, 579, 386

Norris J. P., Bonnell J. T., 2006, ApJ, 643, 266

Norris J. P., Nemiroff R. J., Bonnell J. T., Scargle J. D., Kouveliotou C., Paciesas W. S., Meegan C. A., Fishman G. J., 1996, ApJ, 459, 393

Norris J. P., Marani G. F., Bonnell J. T., 2000, The Astrophysical Journal, 534, 248

Ohno M., et al., 2020, in Holland A. D., Beletic J., eds, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series Vol. 11454, X-Ray, Optical, and Infrared Detectors for Astronomy IX. p. 114541Z (arXiv:2101.05979), doi:10.1117/12.2562253

Pearce T., Brintrup A., Zhu J., 2021, Understanding Softmax Confidence and Uncertainty (arXiv:2106.04972)

Pedregosa F., et al., 2011, J. Mach. Learn. Res., 12, 2825–2830

Pescalli A., et al., 2016, Astronomy & Astrophysics, 587, A40

Pesenson M. Z., Pesenson I. Z., McCollum B., 2010, Advances in Astronomy, 2010, 1

Phung V. H., Rhee E. J., 2019, Applied Sciences, 9

Piran T., 2005, Reviews of Modern Physics, 76, 1143

Pratiwi H., et al., 2020, Journal of Physics: Conference Series, 1471, 012010

Preece R. D., Briggs M. S., Mallozzi R. S., Pendleton G. N., Paciesas W. S., Band D. L., 2000, The Astrophysical Journal Supplement Series, 126, 19

Prochaska J. X., Chen H.-W., Dessauges-Zavadsky M., Bloom J. S., 2007, ApJ, 666, 267

Pál A., et al., 2020, GRBAlpha: A 1U CubeSat mission for validating timing-based gamma-ray burst localization (arXiv:2012.01298)

Pál A., et al., 2023, GRBAlpha: the smallest astrophysical space observatory – Part 1: Detector design, system description and satellite operations (arXiv:2302.10048)

Qi J., Du J., Siniscalchi S. M., Ma X., Lee C.-H., 2020, IEEE Signal Processing Letters, 27, 1485

Raddick M. J., Bracey G., Gay P. L., Lintott C. J., Murray P., Schawinski K., Szalay A. S., Vandenberg J., 2010, Astronomy Education Review, 9

Raschka S., Mirjalili V., 2019, Python Machine Learning. Packt Publishing

Reback J., et al., 2020, pandas, https://pandas.pydata.org/

Ren J., Zhang M., Yu C., Liu Z., 2022, Balanced MSE for Imbalanced Visual Regression (arXiv:2203.16427)

Ricker G. R., et al., 2003, in Ricker G. R., Vanderspek R. K., eds, American Institute of Physics Conference Series Vol. 662, Gamma-Ray Burst and Afterglow Astronomy 2001: A Workshop Celebrating the First Year of the HETE Mission. pp 3–16, doi:10.1063/1.1579291

Ripa J., Dilillo G., Campana R., Galgoczi G., 2020, in den Herder J.-W. A., Nakazawa K., Nikzad S., eds, Space Telescopes and Instrumentation 2020: Ultraviolet to Gamma Ray. SPIE, doi:10.1117/12.2561011

Ripa J., et al., 2023, arXiv e-prints, p. arXiv:2302.10047

Robbins H. E., 1951, Annals of Mathematical Statistics, 22, 400

Rosenblatt F., 1958, Psychological review, 65, 386

Ruby U., Yendapalli V., 2020, International Journal of Advanced Trends in Computer Science and Engineering, 9

Rumelhart D. E., Hinton G. E., Williams R. J., 1986, Nature, 323, 533

Saliwanchik B. R. B., Slosar A., 2022, Publications of the ASP, 134, 114503

Salvaterra R., Campana S., Chincarini G., Choudhury T., Ferrara A., Gallerani S., Guidorzi C., Tagliaferri G., 2008, Proceedings of the International Astronomical Union, 4, 212

Shahmoradi A., 2013, The Astrophysical Journal, 766, 111

Singer S., 1965, Proceedings of the IEEE, 53, 1935

Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., 2014, Journal of Machine Learning Research, 15, 1929

Tanvir N. R., et al., 2009, Nature, 461, 1254

Torigoe K., et al., 2019, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 924, 316

Wang X., Hua Y., Kodirov E., Clifton D. A., Robertson N. M., 2023, IMAE for Noise-Robust Learning: Mean Absolute Error Does Not Treat Examples Equally and Gradient Magnitude's Variance Matters (arXiv:1903.12141)

Werner N., et al., 2018, CAMELOT: Cubesats Applied for MEasuring and LOcalising Transients - Mission Overview (arXiv:1806.03681)

Winkler C., et al., 2003, A&A, 411, L1

Woosley S. E., 1993, ApJ, 405, 273

Zhang Z., Sabuncu M. R., 2018, Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels (arXiv:1805.07836)

Řípa J., Shafieloo A., 2019, Monthly Notices of the Royal Astronomical Society, 486, 3027

van Paradijs J., Kouveliotou C., Wijers R. A. M. J., 2000, Annual Review of A&A, 38, 379