AUSTRIAN ACADEMY OF SCIENCES

INSTITUTE OF HIGH ENERGY PHYSICS

Diplomarbeit

Pulse-Shape Discrimination with Deep Learning in CRESST

Christoph Mühlmann BSc

March 2019

Under Supervision of:
Univ.Prof. Dipl.-Phys. Dr.rer.nat. Jochen Schieck
Univ.Ass. Dipl.-Phys. Dr.rer.nat. Florian Reindl
Univ.Lektor Dr.techn. Wolfgang Waltenberger

TECHNISCHE UNIVERSITÄT WIEN

Abstract

Many unsolved mysteries in our universe such as galaxy rotation curves, mass distribution of clusters etc. can be reasonably explained by the concept of dark matter. WIMPs (Weakly Interacting Massive Particles) are the most favored dark matter candidate. In the quest to experimentally observe WIMP dark matter, CRESST is an outstanding experiment setting the best exclusion limits for low-mass WIMPs ever since. The analysis of the raw data observed by a particular cryogenic detector TUM40 used by CRESST is very challenging as two distinct pulse shapes are observed leading to a two-class classification problem. Neural networks and deep learning evolved to high potential tools in the field of data science. This work uses state-of-the-art deep learning techniques to investigate the two-class classification problem observed in TUM40 data.

Contents

1. Dark Matter

Dark matter is a synonym for different particles that do not interact via the electromagnetic force, hence the name dark matter. The standard model of cosmology (ΛCDM model) is based on cold dark matter (CDM) and the cosmological constant (Λ) together with the cosmological principle which is outlined in this chapter's first two parts. In the third part the main cosmological observations are summarized which indeed hint the existence of non-luminous matter. Furthermore, different models of dark matter are introduced with the prominent one being weakly interacting massive particles (WIMPs) as seen in the fourth part. Finally a brief discussion about the experimental approaches to observe dark matter is given.

1.1. Cosmological Principle

On scales larger than a few Mpc[1] strong indication is given that our universe is isotropic as well as homogeneous. The former statements imply that the universe behaves the same in every spatial direction and observations do not differ in different locations. Both properties combined is denoted as the cosmological principle. A metric that follows the cosmological principle is written as

$$ds^2 = -c^2dt^2 + a(t)^2\left(\frac{dr^2}{1-Kr^2} + r^2d\theta^2 + r^2\sin(\theta)^2d\phi^2\right) \ . \tag{1.1}$$

Equation (1.1) is referred to as the FRW (Friedmann-Robertson-Walker) metric, a(t) denotes the expansion of space as a function of time, $K \in \{-1, 0, 1\}$ corresponds to a hyper-spherical, flat or spherical space respectively and c is the speed of light. The FRW metric together with the Einstein field equations form the following described standard model of cosmology.

1.2. ΛCDM Model

Einstein's field equations are the fundamental equations of general relativity theory. They connect the energy and momentum with the geometry of the universe and are given by

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu}. \tag{1.2}$$

[1] pc (parsec) is defined as the distance between an arbitrary point on the line originating from the sun perpendicular to the rotation plane of the earth around the sun where the average distance between the earth and the sun is at an angle of one arcsec measured at that arbitrary point. A pc is approximately $3{\cdot}10^{16}$ meters.

Solving Einstein's equation together with the FRM metric given by (1.1) leads to the following fundamental Friedmann equations[2] where the first one reads

$$\left(\frac{\dot{a}(t)}{a(t)}\right)^2 + \frac{c^2 K}{a(t)^2} - \frac{c^2\Lambda}{3} = \frac{8\pi G}{3}\rho(t) \ . \tag{1.3}$$

The first part of equation (1.3) is defined as the Hubble constant $H(t)^2 := \left(\frac{\dot{a}(t)}{a(t)}\right)^2$, $\rho(t)$ is the energy density of the universe, G is the gravitation constant and Λ is the cosmological constant. It is common to transform $\rho \to \rho$ - $\Lambda/8\pi$G and set K to zero to evaluate the critical energy density as

$$\rho_c(t) = \frac{3H(t)^2}{8\pi G}. \tag{1.4}$$

As radiation, matter and the cosmological constant[3] contribute to the energy density ρ the former different parts are normalized as $\Omega_i = \rho_i/\rho_c$. Based on this normalization together with equation (1.4) equation (1.3) can be rewritten in terms of

$$\Omega_T(t) = \sum_i \Omega_i(t) = 1 + \frac{c^2 K}{\dot{a}(t)^2}. \tag{1.5}$$

As discussed, the total energy density of our universe seen in equation (1.5) consists of the following three parts

$$\Omega_T = \Omega_{matter} + \Omega_{radiation} + \Omega_\Lambda \ . \tag{1.6}$$

The contribution from matter can be further divided into

$$\Omega_{matter} = \Omega_{baryons} + \Omega_{leptons} + \Omega_{neutrinos} + \Omega_{dark\ matter}. \tag{1.7}$$

In summary the ΛCDM model is derived with the cosmological principle and the FRW metric combined with the Einstein field equations. The energy density of the universe is made up of matter, radiation and the contribution from the cosmological constant. Furthermore, the energy density given by matter can be further divided with one part consisting of cold dark matter. An overview of the main different contributions to our universe is seen in figure 1.1.

1.3. Evidence for Dark Matter

Many cosmological as well as astrophysical observations on all cosmological scales hint the existence of matter that is non-luminous. The following part briefly summarizes prominent observations.

[2]The derivation of the Friedmann equations leads to two equations. For the considerations of this chapter only the first one is needed.

[3]The influence of the cosmological constant Λ on a(t) is usually referred to as dark energy.

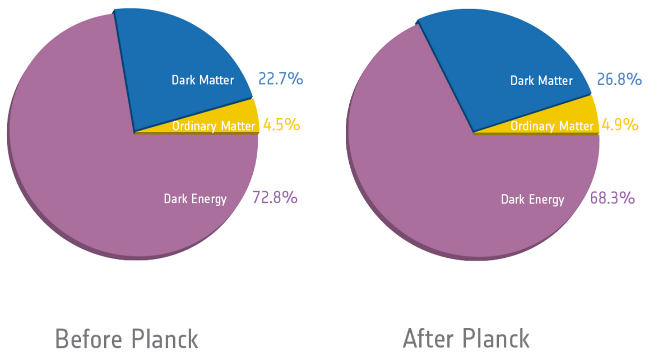


Figure 1.1.: Energy density contributions to our present universe based on nine year WMAP data (left chart) and Planck data (right chart). This illustration is taken from [1].

Cosmic Microwave Background

In 1978 Arno Penzias and Robert Wilson were awarded with the Nobel Prize in Physics for the discovery and related work regarding the cosmic microwave background (CMB) [2]. The origin of this radiation lies about 380,000 years after the big bang. At that time the expansion of the universe and the resulting decrease in temperature caused a suppression of the disintegration process of hydrogen with photons. Hence photons were able to travel freely (photon decoupling) and are observed all over the sky with a nearly perfect black body radiation spectrum. Due to the expansion of the universe over time the wavelengths of the photons are increased which leads to a red-shift of the CMB resulting in a temperature of $2.72548 \pm 0.00057K$ [3]. Another red-shift of the CMB originates by an inhomogeneous mass density of the universe at the time of photon decoupling. The former temperature fluctuations are observed by many experiments including WMAP [4] and Planck [5]. Given these temperature fluctuations many cosmological quantities based on the ΛCDM model can be constrained. In addition, it can be derived that roughly a quarter of the energy density of our universe is indeed made up of dark matter. An overview of the findings based on nine year WMAP and Planck data can be seen in figure 1.1.

Virial Theorem applied to Galaxy Clusters

In classical mechanics the virial theorem connects the mean potential energy with the mean kinetic energy of a N-body problem. F. Zwicky applied the virial theorem to observations of the COMA cluster and found that the observed objects move to fast which can be explained by additional mass which does not interact with light [6].

Galaxy Rotation Curves

A galaxy rotation curve is a depiction of the angular velocities of the respective stars as a function of their radial distance measured from the galaxy center. Such a curve is expected to rise linearly with the distance from the center of the galaxy r until a maximum is achieved and then decrease proportional to $1/\sqrt{\mathrm{r}}$. This fact is derived by the equality of the absolute value of the gravitational force and the centrifugal force. However, many observed galaxies including the Milky Way do not show a decrease but rather a constant angular velocity [7, 8]. An explanation of this behaviour can be made with an additional presence of dark matter which does not interact with light.

Mass Distribution of the Bullet Cluster

The Bullet Cluster consists of two galaxy clusters traversing each other. While the X-ray emitting gas components experience friction the remaining masses (stars, dark matter) move by freely. By measuring the X-rays as well as the mass distribution a spatial partitioning is observed. Contrary to the measurements the center of mass is expected in the gas components as $\mathrm{m_{Gas}} \gg \mathrm{m_{Stars}}$. This discrepancy can be explained via dark matter present in both galaxies traversing without friction and is outlined in detail in [9].

1.4. Dark Matter Candidate Theories

In principle observations that indicate the existence of dark matter can be explained by many theories. Three distinct approaches are outlined as follows.

MOND

MOND (Modified Newtonian Dynamics) is a theory which extends the law of gravitation without introducing a new particle, furthermore, the considerations outlined in section 1.2 would be violated because the ΛCDM model is based on particle dark matter. However, this theory does not account for mechanisms such as the CMB fluctuations, the Bullet Cluster measurements and others as described in [10].

MACHO

MACHOs (Massive Astrophysical Compact Halo Objects) describe non-luminous baryonic matter. Experiments based on gravitational lensing observed MACHOs but it is excluded that MACHOs account for all observed dark matter mass as described in [11, 12].

Unknown Particles

A promising approach is to describe dark matter in terms of non-baryonic unobserved particles as all Standard Model particles are excluded. Many hypothetical particles fulfill the necessary requirements as described in detail in [13, 14]. One well motivated

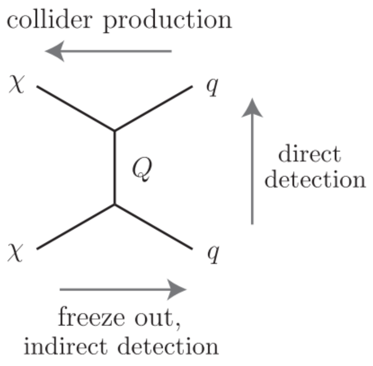


Figure 1.2.: Feynman diagram of dark matter particles interacting with standard model particles. Three distinct processes which are used to experimentally search for dark matter can be seen. This illustration is taken from [15].

candidate is the WIMP (Weakly Interacting Massive Particle) which is proposed to have a mass of typically weakly interacting particles. In addition, WIMPs interact only via the gravitational force as well as a weak force with roughly the same strength as the weak nuclear force. The density of dark matter today can be derived via thermodynamic laws, this derivation needs an interaction cross section as well as a dark matter particle mass as inputs. If the former calculation is carried out with typical values for the cross section and mass of weakly interacting particles, the measured dark matter density is derived. This circumstance is referred to as the WIMP miracle and further increases the interest in WIMPs as candidates for dark matter.

1.5. Dark Matter Detection

Dark matter particles might interact with ordinary standard model particles as illustrated in the Feynman graph seen in figure 1.2. Based on this interaction scheme three distinct approaches are capable of searching for dark matter. Indirect detection experiments measure the processes of dark matter particles decaying or annihilating into standard model particles. Collider production experiments aim to produce dark matter from standard model particles and infer dark matter via missing energy and momenta based on the production processes. And lastly direct detection experiments measure nuclear recoils[4] induced by dark matter particles based on the expected dark matter interaction rate for different models. CRESST is an experiment searching for WIMPs by the direct detection principle, hence the WIMP interaction rate is needed and briefly outlined as follows.

Expected WIMP Differential Recoil Spectrum

Assuming that WIMPs are present in the Milky Way, nuclear recoils occur due to the movement of the earth throughout the galaxy. Based on this assumption the differential

[4]Recently also electron recoil dark matter searches evolve.

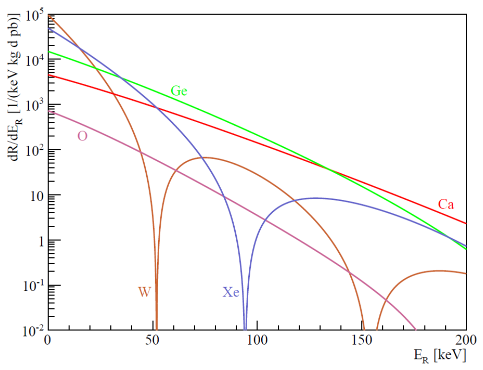


Figure 1.3.: Depiction of the differential recoil spectra as a function of the recoil energy for different target materials. For this plot a WIMP mass of 100 GeV/c^2 is considered. This illustration is taken from [16].

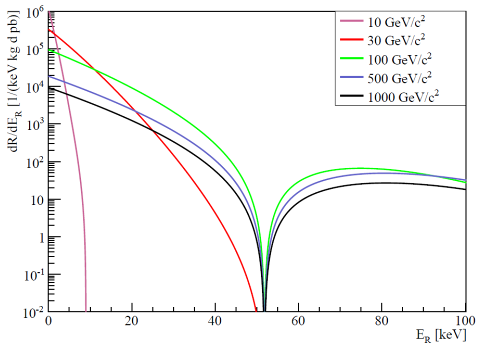


Figure 1.4.: Illustration of the differential recoil spectra as a function of the recoil energy for different WIMP masses. For this plot tungsten is used as target material. This illustration is taken from [16].

recoil spectrum is a product of three distinct terms as follows. The first term is the density of the target nuclei in the detector. The second term is the flux of the incoming WIMPs which is dependent on the WIMP velocity. Two effects have to be considered for the WIMP velocity distribution. Firstly an upper bound is given by the galactic escape velocity, WIMPs that are faster are not bound to the Milky Way, and secondly an annual modulation originating from the movement of the earth around the sun has to be considered. The third term represents the differential WIMP scatter cross section. This term is proportional to the squared target mass number A and a form factor that accounts for the fact that at high momentum transfers the target cannot be approximated as a point. A detailed discussion of the WIMP differential recoil spectrum can be found in [16]. In figure 1.3 the differential recoil spectra for different target materials and for a WIMP mass of 100 GeV/c^2 are depicted. Figure 1.4 shows the differential recoil spectra for different WIMP masses and tungsten as target material. Based on both figures, a low detector threshold is of utmost importance to ensure a high WIMP interaction rate. This circumstance is amplified as the WIMP mass decreases. Therefore, CRESST aims for a very low detector threshold. The experiment itself as well as the detector principle is outlined in the following chapter.

2. CRESST

CRESST (Cryogenic Rare Event Search with Superconducting Thermometers) is an experiment dedicated to hunt dark matter based on nuclear recoil processes, it is located in the Laboratori Nazionali del Gran Sasso (LNGS) in Italy. Starting in 1995, CRESST developed over three stages with different detector materials and designs setting new limits on the cross-section of WIMP dark matter ever since. A summary of the different CRESST stages and phases as well as the main publications is seen in table 2.1. The unique two-channel detector design based on the phonon and a scintillation light signal provides a method to distinguish between different types of particles involved in an interaction, thus this design delivers an active background discrimination. CRESST-II and CRESST-III are based on these two channel cryogenic detectors. CRESST-I used only one channel to measure the phonon signal. This work focuses solely on CRESST-II phase 2 which is further denoted as CRESST. This chapter will give an general idea of the experimental setup leading into an introduction of the two-channel detector setup and finishing with the fundamentals of the raw data preparation and selection for CRESST detectors.

2.1. Experimental Setup

Figure 2.1 depicts the experimental setup of the CRESST experiment. A large fraction of the experiment is dedicated to the shielding against various sources of background such as Muons, Gamma Quanta, Electrons, Radon and Neutrons. In addition, LNGS provides 1400 m of rock which is equivalent to approximately 3600 m of water [25]. In the upper region of figure 2.1 the cooling complex is seen which provides temperatures of approximately 5 mK by the principle of He^3/He^4 dilution. The heart of the experiment holds the cryogenic detectors which are described in the following chapter. A detailed description of CRESST is seen in [16, 18].

Table 2.1.: Summary of different CRESST stages and phases as well as the respective main publications.

Stage	Phase	Data Taking	Publications
CRESST-I	-	2000	[17]
	Commissioning	2007	[18, 19]
CRESST-II	Phase 1	2009-2011	[20]
	Phase 2	2013-2015	[21, 22]
CRESST-III	Phase 1	$\geq$2016	[23, 24]

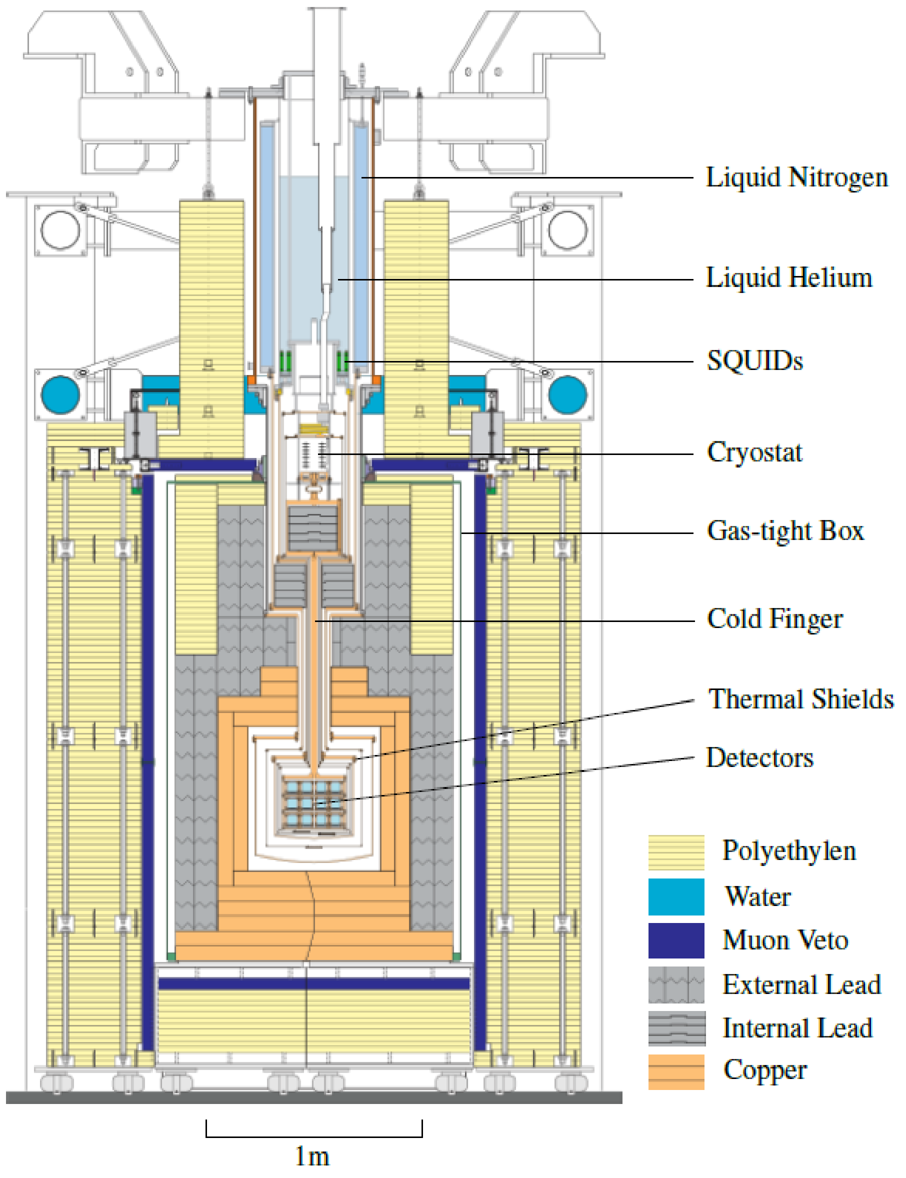


Figure 2.1.: Illustration of the CRESST experimental setup. This illustration is taken from [26].

2.2. Two-Channel Cryogenic Detector Design

CRESST uses two-channel cryogenic detectors as seen in figure 2.2. Energy deposited by particles in the target is measured in two channels. Firstly a phonon signal which is measured in one channel and secondly scintillation light that is measured in the other channel. The signals of both channels combined deliver a background discrimination method. In the first part a general introduction of the detector concept is given. This chapters second part concentrates on a physical model to describe the measured signal shape.

2.2.1. Detector Working Principle

The core component of a two-channel cryogenic detector seen in figure 2.2 is the target crystal. Nuclear recoils induced by incoming particles deposit energy in the crystal which leads to crystal lattice vibrations. From a physical standpoint this lattice vibrations are phonons, thus a temperature increase is caused by any particle interaction. Based on thermodynamics the energy increase is approximately connected with the temperature increase as follows

$$\Delta T = \frac{\Delta E}{C} \ . \tag{2.1}$$

In equation (2.1) C denotes the heat capacity of the target crystal. This quantity depends on temperature. For temperatures much smaller than the Debye temperature C scales with T^{-3}, hence at very low temperatures a tiny energy deposition leads to a rather high temperature increase of the target crystal. As the former equation is only an approximation a detailed discussion of the temperature rise induced by a particle interaction is seen in section 2.2.2. In addition to the phonon signal, particle interactions also produce scintillation light. The detector is surrounded by a reflective housing that guides the scintillation light into the light absorber which is another crystal that gets heated by the incoming photons. TESs (Transition Edge Sensors) are mounted on the target crystal and the light absorber respectively, thus the crystal/light absorber and the respective TES are in direct thermal contact. Those TESs are type-I superconductors operated in the transition region. Due to the temperature increase of a crystal originating from a particle interaction the TES heats up and its resistance changes as illustrated in figure 2.3. This increase of the resistance is finally measured with a SQUID readout resulting in a measured signal for the phonon as well as the light channel independently. CRESST uses mostly $CaWO_4$ as target material for the crystals. A typical event measured in the phonon channel is seen in figure 2.5.

Background Discrimination

The amount of produced scintillation light ($<10\%$ for $CaWO_4$) strongly depends on the type of the interacting particle, whereas the phonon signal is quasi independent of the particle type. Gammas and electrons interact with the electrons in the crystal via the electromagnetic force, hence more scintillation light is produced compared to a recoil

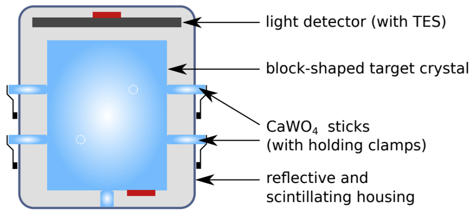

light detector (with TES)
block-shaped target crystal
CaWO$_4$ sticks (with holding clamps)
reflective and scintillating housing

Figure 2.2.: Depiction of a CRESST two-channel cryogenic detector design. This illustration is taken from [27].

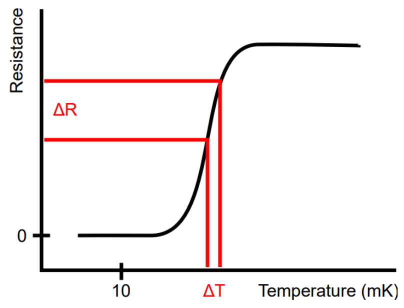


Figure 2.3.: Illustration of a transition curve for a Type-I superconductor which is used to measure the temperature rise of CRESST detector target crystals.

process originating from a neutral particle. Based on the photon and the light signal the light yield parameter is defined as

$$LY = \frac{E_{Scintillation}}{E_{Phonon}} . \tag{2.2}$$

Processes originating from electrons or gammas produce the most amount of scintillation light, consequently the light yield is normalized to one for electron/γ processes. This is done by setting the observed phonon energy equal to the light channel energy for electron/γ events at a specific calibration energy, resulting in a light yield parameter of one for electron/γ events at that specific calibration energy. Particle interactions with less observed scintillation light produce light yields smaller than one. A typical light yield plot for a CaWO$_4$ crystal used in CRESST is seen in figure 2.4. Following the discussion in [16] the WIMP nuclear recoil cross section scales with the squared mass number A of the target nuclei. Considering CaWO$_4$ crystals WIMP events are mostly expected in the tungsten band. However, for low WIMP masses the deposited energy can be lower as the detector threshold for tungsten recoils. This would lead to a shift into the oxygen band for the observable particle interactions. The same argumentation holds for neutrons, resulting in the bulk of neutron background events observed in the oxygen band. Based on the former discussion the light yield parameter provides a high[1] background discrimination as seen in figure 2.4.

2.2.2. Pulse Shape Model

Figure 2.5 illustrates a measured phonon signal. The underlying physical model behind the pulse shape is of great interest. A detailed discussion of the pulse shape seen in

[1]For CRESST-II detectors a background reduction of at least $1/10^5$ at 46 keV is provided by the two channel detector design [20].

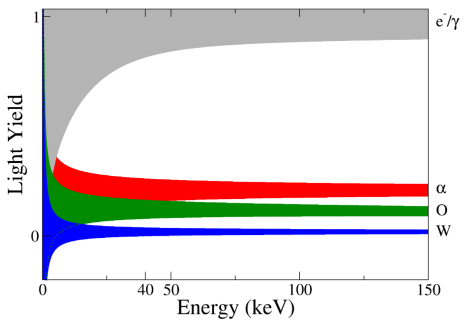


Figure 2.4.: Depiction of a light yield plot for a $CaWO_4$ target crystal. This plot is taken from [28].

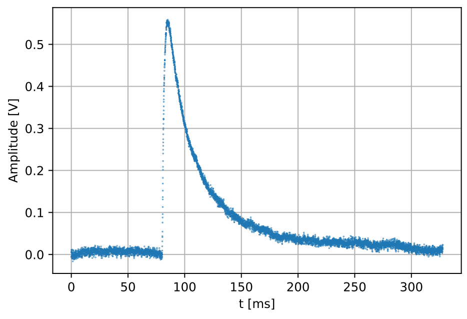


Figure 2.5.: Measured phonon signal with a fit energy of 35 keV. The amplitudes correspond to the SQUID outputs.

cryogenic detectors used in CRESST is given by [29]. The main points of this publication will be summarized in the following parts.

General Considerations

A particle interaction in the target crystal leads to initial phonons of very high frequencies. These unstable phonons decay with a rate that is proportional to the fifth power of their frequency ($\Gamma \propto \nu^5$) with an initial frequency that is approximately half the Debye frequency of the detector material. Due to the frequency-dependent decay rate of the phonons a rapid average frequency decrease is observed which is followed by a much slower decay rate resulting in an approximately constant frequency on time scales of micro seconds. This constant frequency is much higher than the average frequency of the thermal phonons, thus the initial phonons are non-thermal. Combining that the crystal diameter is in the order of a few centimeters and the typical speed of sound is roughly 10^3 m/s leads to the fact that the non-thermal phonons immediately (μs) fill the target crystal. As a result the TES which is mounted directly on the target crystal is as well immediately filled with non-thermal phonons. Electrons in the TES efficiently absorb those phonons which leads to a rise in the temperature of the TES electron system and furthermore resulting in a change of the TES resistance. In summary the former statements lead to a fast power input in the TES electron system $P_e(t)$ that is responsible for the fast acting component of the measured signal.

The bulk of the heat in the TES electron system originating from the non-thermal phonons escapes into the heat bath and does not flow back into the absorber as the thermal coupling of the TES electron system with the absorber is very small at low temperatures. A power input to the thermal phonons $P_a(t)$ is given by the fact that a part of the non-thermal phonons thermalizes on the surface of the target crystal. This power input results in a slow acting component of the measured signal.

According to the outline above an initial particle interaction results in two power inputs for the TES electrons system as well as the target crystal phonon system respectively.

In order to derive a pulse shape the phonon frequency dependency of the power inputs is neglected and the heat flow in the whole detector is evaluated based on the thermal model of the whole detector design seen in figure 2.6. The change in temperature of the TES electron system determines the change in resistance of the TES which is the measured signal, hence $\Delta T_e(t)$ will be calculated. A particle depositing energy ΔE into the crystal leads to the following power inputs

$$P_e(t) = \Theta(t) P_0 e^{-t/\tau_n}, \qquad P_a(t) = \frac{1-\epsilon}{\epsilon} P_e(t) \qquad \text{with} \qquad P_0 = \frac{\epsilon \Delta E}{\tau_n} \ . \tag{2.3}$$

In equation (2.3) P_0 is the initial non-thermal phonon power input that splits into the TES electron system and the target crystal phonon system according to ϵ. The initial power input P_0 decays with a time constant τ_n in terms of the two processes which are discussed above, namely the thermalization of the phonons on the crystal surface represented by τ_{Crystal} and thermalization of the phonons given by the TES electron system represented by τ_{TES} leading to

$$\tau_n = \left(\frac{1}{\tau_{TES}} + \frac{1}{\tau_{Crystal}} \right)^{-1} \ . \tag{2.4}$$

The form of the two power inputs seen above together with the model seen in figure 2.6 are used to describe the temperature change of the TES electron system described in the following part.

Derivation of the TES Electron Temperature Response

Two differential equations can be formulated by considering the thermal model seen in figure 2.6 as well as the power inputs defined in equation (2.3) and additionally neglecting the spatial dependence of the TES electron system temperature. Leading to the following equations

$$C_e \frac{dT_e}{dt} + (T_e - T_a) G_{ea} + (T_e - T_b) G_{eb} = P_e(t) \tag{2.5}$$

$$C_a \frac{dT_a}{dt} + (T_a - T_e) G_{ea} + (T_a - T_b) G_{ab} = P_a(t) \ . \tag{2.6}$$

Equation (2.5) and (2.6) can be analytically solved with the initial conditions that the temperature of the target crystal as well as the temperature of the TES electron system is equal to the temperature of the heat bath. The solution is given by

$$\Delta T_e(t) = \Theta(t - t_0)[A_n(e^{-(t-t_0)/\tau_n} - e^{-(t-t_0)/\tau_{in}}) + A_t(e^{-(t-t_0)/\tau_t} - e^{-(t-t_0)/\tau_n})] \ . \tag{2.7}$$

In equation (2.7) $\Delta T_e(t)$ is defined as $T_e(t)$-T_b. As $C_e \ll C_a$ holds true for the considered detector design the decay times seen in equation (2.7) evaluate as

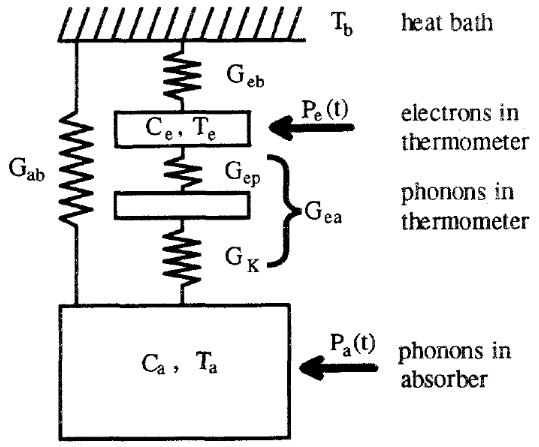


Figure 2.6.: Illustration of the thermal model of a cryogenic detector design which is described in chapter 2.2.1. $P_a(t)$ denotes the power input to the thermal phonons in the target crystal, T_a is the temperature of the target crystal, C_a is the heat capacity of the target crystal. $P_e(t)$ denotes the power input to the TES electron system, T_e is the temperature of the TES electron system, C_e is the heat capacity of the TES electron system. G_{eb} describes the thermal conductance between the TES electron system and the heat bath. G_{ep} is the thermal conductance between the TES electron system and the TES phonon system, G_K is the thermal conductance between the TES phonon system and the target crystal. The former two quantities define G_{ea} which is the resulting thermal conductance between the TES electron system and the target crystal derived by an inverse sum. G_{eb} denotes the thermal conductance between the TES electron system and the heat bath which has a temperature of T_b. Lastly G_{eb} denotes the thermal conductance between the target crystal and the heat bath. This illustration is taken from [29].

$$\tau_{in} \approx \frac{C_e}{G_{ea} + G_{eb}} \qquad \tau_t \approx \frac{C_a}{G_{eb}G_{ea}/(G_{eb} + G_{ea}) + G_{ab}} \tag{2.8}$$

and the amplitude of the non-thermal component reads

$$A_n \approx \frac{P_0}{(G_{eb} + G_{ea})(1 - \tau_{in}/\tau_n)(1 - \tau_{in}/\tau_t)} = \frac{-\epsilon \Delta E}{C_e(1 - \tau_n/\tau_{in})(1 - \tau_{in}/\tau_t)} \ . \tag{2.9}$$

The sign of the amplitude of the non-thermal component given by equation (2.9) is determined by the ratio of $\tau_{\mathrm{in}}/\tau_{\mathrm{n}}$, thus two distinct detector operation modes can be utilized as follows.

$\tau_{\mathrm{in}} \ll \tau_{\mathrm{n}}$: In this case τ_{in} is the rise time and τ_{n} is the decay time seen in equation (2.7) which leads to $\Delta T_e(t) \propto P_0 e^{-(t-t_0)/\tau_n}$. The temperature rise of the TES electron system has the same shape as $\mathrm{P_e(t)}$ which is seen in equation (2.3). As a result this operation mode measures the time dependent non-thermal phonon flux and is referred to as the **bolometric mode**.

$\tau_{\mathrm{n}} \ll \tau_{\mathrm{in}}$: In this case the amplitude of the non-thermal component seen in equation (2.7) is proportional to the energy of the high frequency phonons given by $A_n \approx \frac{-\epsilon \Delta E}{C_e}$. Thus this mode is referred to as the **calorimetric mode** because the total energy of the high frequency phonons is measured.

In the former introduced differential equations (2.5) and (2.6) it is assumed that the electron system of the TES is a perfect thermal conductor. However, $\mathrm{G_{TES}}$ also denoted as $\mathrm{G_{Film}}$ is smaller compared to $\mathrm{G_{eb}}$ which is also denoted as $\mathrm{G_{Au}}$. To take this effect into account the temperature distribution of the TES electron system has an additional dependency on the spatial coordinate $\mathrm{T_e = T_e(x,t)}$. Correcting equations (2.5) and (2.6) for this effect will lead to coupled partial differential equations of second order which cannot be solved analytically. After a rather long calculation equation (2.7) still holds true but with adapted decay times and amplitudes given by

$$\tau_t = \frac{C_a}{\gamma G_{ea} + G_{ab}} \qquad \tau_{in} = C_e \frac{(1 - \gamma)}{G_{ea}} \tag{2.10}$$

$$A_t = \frac{\Delta E}{C_a} \frac{(1 - \gamma)(1 - \epsilon\gamma)}{(1 - \tau_n/\tau_t)} \qquad A_n = \frac{(1 - \gamma)}{G_{ea}(1 - \tau_{in}/\tau_n)} P_0 \tag{2.11}$$

$$\gamma = \left(\frac{G_{ea}}{G_{Au}} + \frac{\lambda L}{\tanh \lambda L} \right)^{-1} \qquad \lambda L = \sqrt{\frac{G_{ea}}{G_{film}}} \qquad G_{film} = \frac{\kappa V}{L^2} \ . \tag{2.12}$$

In equation (2.12) κ, V and L denote the thermal conductivity of the film, the volume of the film and the length of the film respectively. Equation (2.7) together with equations (2.4), (2.10), (2.11) and (2.12) form the final model for the pulse shape of a cryogenic detector. In summary the pulse shape model seen in equation (2.7) has two distinct components, namely a fast component originating from the initial non-thermal phonons and a slow component resulting from the decay of the non-thermal component. Therefore, the decay of the fast component equals the rise of the slow component that is determined by the time constant τ_n. τ_t characterizes the decay of the thermal component and is mainly determined by the thermal conduction between the TES electron system and the heat bath as well as the thermal conduction between the target crystal and the heat bath. Furthermore, it is observed that the approximate temperature change given by equation (2.1) is measured by the thermal component but corrected as seen in equation (2.11). This correction partly accounts for the fact that some of the heat from the non-thermal phonons escapes into the heat bath. Lastly the resulting pulse shape model is independent of the interacting particle type resulting in the same pulse shape for every particle and pulses only differ by the deposited energy in terms of an almost linear change in the amplitude. A fit of the introduced pulse shape model given by equation (2.7) on real measured pulses is depicted in figure 3.3 and discussed in section 3.1.1.

2.3. Raw Data Preparation

The former chapter emphasized the detector working principle and introduced a physical model to describe the pulse shape measured with cryogenic detectors. This chapter focuses on the practical considerations in analyzing measured pulses. Figure 2.5 depicts a measured phonon channel signal. Technically the voltage output of the SQUID is measured every 0.04 ms, every voltage measurement is referred to as a sample. An event is formed by 8192 samples resulting in a record length of about 328 ms. In the next chapter a selection of simple quantities that are calculated for every record are introduced.

2.3.1. Basic Pulse Parameters

Many pulse parameters are measured and calculated for all observed pulses in the whole data analysis process as seen in [26, 30]. For the aim of the present work only three parameters are needed which are described as follows.

Pulse Height is determined by firstly applying a 50 sample running average on the pulse. Secondly the average of the first 50 resulting samples is evaluated and subtracted from every sample. Lastly the maximum sample is determined as the pulse height parameter.

10% - 70% Rise Time is the time difference between the 70% and 10% sample relative to the maximum sample.

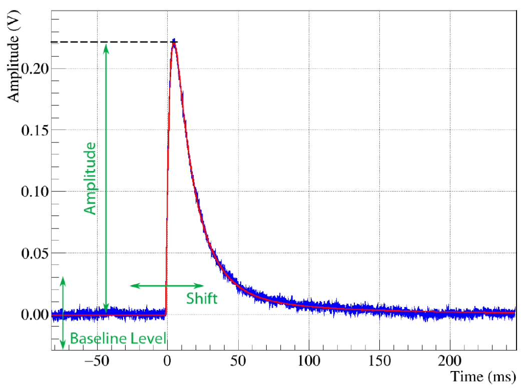


Figure 2.7.: Depiction of the standard event fit with its three distinct fitting parameters. This illustration is taken from [30].

Decay Time is the time between the maximum sample and the sample corresponding to 1/e of the pules height.

The pulse height parameter is a first estimation of the amplitude of the pulse. However, as seen in figure 2.5 the pulse is superimposed with noise that is somewhat averaged out by the 50 sample moving average. In general, the pulse height parameter is biased towards higher values, thus, a superior approach to determine the amplitude of the pulse is described in the following chapter.

2.3.2. Template Fit

The amplitude of a measured pulse can be evaluated by fitting a template. This template or also referred to as a standard event is as well a record consisting of 8192 samples. It is obtained by averaging every sample of many measured records with equal amplitudes and then it is scaled to an amplitude of 1 V, thus the variance of the (random) noise should be averaged out and it resembles the typical shape of a pulse. Figure 2.7 illustrates the fitting procedure with its three free parameters. The template can be scaled which is done by multiplying every sample with a scale parameter, furthermore the template can be shifted in time to correct for events that have a time offset. Lastly a baseline offset can be introduced by adding a value to every sample. By minimizing the MSE (Mean Squared Error) the former three parameters are obtained[2] with the scale factor corresponding to the amplitude. This fitting procedure can be extended in several ways in order to adapt to the detailed structure of the data which is among others discussed in [30].

[2]Note that this fitting procedure without the time shift is a linear fit, thus the solution can be given in a closed form.

2.3.3. Energy Evaluation

After evaluating the amplitude of a record with the standard event fit described above the deposited particle energy can be reconstructed. As known from the discussion in chapter 2.2.2 the amplitude of the event scales linearly with the deposited particle energy[3]. For the purpose of the energy reconstruction an electrical heater is mounted on every target crystal. The heater current circuit is designed such that the deposited heating energy scales linearly with the injected heater voltage. A heater pulse standard event can be obtained and the standard event fitting procedure can be applied as described in the former chapter. Based on heater pulses the energy reconstruction of a real pulse is evaluated in two steps as described in a simplified version as follows.

Energy Calibration

For the energy calibration at time $\mathrm{t_0}$ heater pulses with one particular voltage $\mathrm{V_0}$ are injected into the detector[4]. The observed heater pulses are fit with the heater pulse standard event resulting in an amplitude $\mathrm{A_{h,0}}$. Additionally, the detector is irradiated with a ^{57}Co source which has a prominent gamma peak at 122.1 keV. Many events originating from this source are used to obtain the standard event. Events originating from the known source deliver a fit amplitude $\mathrm{A_{Co,0}}$ with a known deposited energy of $\mathrm{E_{Co}}$. Heater pulses scale linearly with the injected heater voltage and additionally at zero injected voltage an amplitude of zero is observed. Based on those considerations a line can be defined as follows

$$V_{h,0}(A) = \frac{V_0}{A_{h,0}} A \ . \tag{2.13}$$

In order to connect the energy of the known source with the injected heater voltage the CPE (Convert Pulse-Height to Energy) factor is defined as follows

$$CPE = \frac{E_{Co}}{V_{h,0}(A_{Co,0})} = \frac{1}{V_0} \frac{A_{h,0}}{A_{Co,0}} E_{Co} \ . \tag{2.14}$$

The CPE factor connects the injected heater voltage with a corresponding deposited particle energy, for example an injected heater voltage of 1 V is equivalent to a electron/γ interaction depositing an energy of 1V$\cdot$CPE. This factor as seen in equation (2.14) is usually determined once for every detector in a data taking phase. Additionally, heater pulses are injected constantly over the whole data taking phase to monitor the detector response as a function of energy and time. Furthermore, control pulses which are heater pulses that drive the TES completely out of the transition region are injected to measure and stabilize the detector working point.

[3]This statement is only true if the temperature rise lies in the linear region of the TES transition curve depicted in figure 2.3.

[4]The injected heater voltage is the applied voltage to the heater current circuit. It must not be mistaken with the amplitudes delivered by the standard event fit which are given in Volts as well.

Energy Reconstruction

In this simplified explanation the detector response is measured constantly with an injected heater voltage of $V_t = V_0$. The amplitude $A_{h,t}$ for the injected heater pulses at time t is determined with the heater standard event fit. If the detector response has changed compared to energy calibration time the measured amplitude will change as well even if the same heater voltage is injected as done in the calibration phase. In analogy to the calibration step a line at time t can be defined as follows

$$V_{h,t}(A) = \frac{V_t}{A_{h,t}} A \ . \tag{2.15}$$

A particle event happening approximately at time t will be fit with the particle standard event resulting in an amplitude $A_{p,t}$. The CPE factor determined at calibration time in conjunction with the detector response at time t given by equation (2.15) is used to reconstruct the particle energy with the following equation.

$$E_p = V_{h,t}(A_{p,t}) \cdot CPE = \frac{A_{p,t}}{A_{Co,0}} \frac{A_{h,0}}{A_{h,t}} E_{Co} \ . \tag{2.16}$$

The first two terms of the right part seen in equation (2.16) can be interpreted as follows. The first fraction accounts for the fact that different deposited energies have a different amplitude, the second fraction adjusts for changes in the detector response function. If the fit heater amplitude increases over time given the same injection voltage the energy would be overestimated. However, as the second fraction would be smaller than one in that situation the overestimation is suppressed.

This outlined procedure of energy calibration as well as energy reconstruction is done for the light and for the phonon channel respectively, therefore two CPE factors are determined and used for the energy reconstruction. At calibration the energy in the phonon and the light detector are set to the energy of the calibration source resulting in a light yield of one at a deposited energy of the calibration source (122.1 keV for ^{57}Co). As a γ-source is used this energy reconstruction implicitly assumes that every measured event is a γ with a certain energy fraction η emitted as scintillation light. For events that induce less/more scintillation light the energy reconstruction overestimated/underestimates the total deposited energy, therefore the following correction is applied to the reconstructed phonon energy E_p to derive E which is the total particle independent deposited energy[5].

$$E = \eta E_l + (1 - \eta) E_p = (1 - \eta(1 - LY)) E_p \tag{2.17}$$

The above outlined energy calibration and reconstruction procedure is simplified in order to explain the intuition behind it. Usually heater pulses of many different energies are injected and fit with a polynomial of higher order rather than a line without an offset. Figure 2.8 shows a measured detector response curve with the energy reconstruction for the TUM40 detector. It is observed that the polynomial fit to the heater pulses is very

[5]For this circumstance the reconstructed phonon energy E_p is given in keVee (electron equivalent) and the total energy E is given in keV.

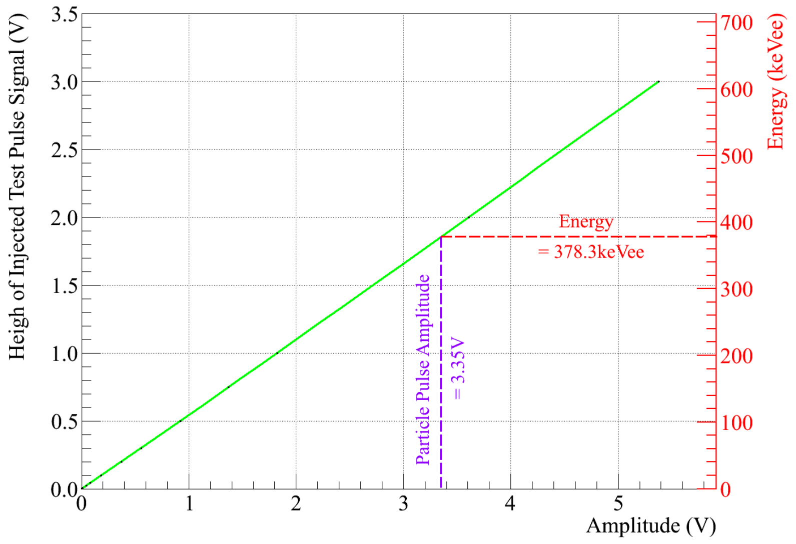


Figure 2.8.: Illustration of a TUM40 cryogenic detector response curve. The abscissa correspond to the amplitude given by the standard event fit for heater as well as particle pulses. The left ordinate represents the injected heater voltage and the right ordinate is the evaluated electron equivalent energy based on the CPE factor. This illustration is taken from [30].

close to linear at time of the measurement, therefore a fit particle event amplitude of one volt corresponds to approximately 111 keVee. For $CaWO_4$, which is the target material of TUM40, η=6.6% [21], the TUM40 detector is described in detail in section 3. More technical details regarding the energy calibration and reconstruction are seen in [26, 30].

3. Cryogenic Detector TUM40

Chapter 2.2 introduced the general working principle and the design of two-channel cryogenic detectors. In CRESST detector modules with a composite design and $CaWO_4$ as target material were used among others, this composite detector design follows the same outlined working principle but differs in on detail in the construction which leads to an additional pulse shape. The main results of these modules are outlined in [21, 22], a detector threshold of 307 eV was achieved which enabled setting limits on light dark matter that where state-of-the-art at that time. One particular composite detector module is the TUM40, this work exclusively focuses on data measured by this detector. In the following parts the composite detector design is introduced. Based on the two observed pulse shapes a classification problem arises. Several approaches for the pulse shape discrimination are discussed.

3.1. Composite Detector Design

In a standard detector design, also referred to as conventional detector design, the TES gets evaporated on the target crystal, however the high temperature exposure on $CaWO_4$ leads to an oxygen deficit which decreases the scintillation light output. To maximize the light output the TES is evaporated on a small $CaWO_4$ crystal labeled as carrier. This carrier is then glued on the target crystal that is labeled as absorber. A sketch of the core composite detector design is seen in figure 3.1 and a real photograph of the target of the TUM40 detector is seen in figure 3.2. This outlined detector design is referred to as composite detector design and is introduced in [31]. Additionally to the improved light output, it has the disadvantage that particle interactions can occur in the absorber as well as in the carrier as discussed in the following chapter.

3.1.1. TUM40 Pulses Shapes

For the TUM40 detector the absorber standard event is generated with events originating from the known ^{57}Co source and the carrier standard event is generated with many carrier events that have the same amplitude. In section 2.2.2 the cryogenic detector pulse shape model was introduced with the resulting model seen in equation (2.7). The former pulse shape is fit to both standard events which is illustrated in figure 3.3, and the resulting fit parameters are summarized in table 3.1. As equation (2.7) is non linear in its parameters the fitting procedure has no closed form solution, hence the basin hopping non linear solver was used. This algorithm is based on simulated annealing with an additional step of gradient decent at every jump in the fit parameter space. Based on figure 3.3 it is observed that the carrier pulse is fast rising and fast decaying due to the direct

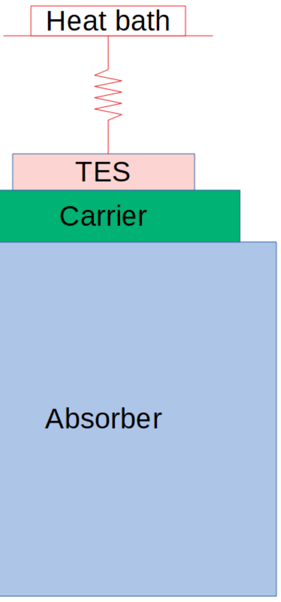


Figure 3.1.: Schematic of the target crystal of the composite detector design. The thermometer is evaporated on the carrier crystal which is glued on the absorber crystal. This illustration is not to scale. The holding clamps, light detector and the detector housing are not depicted.

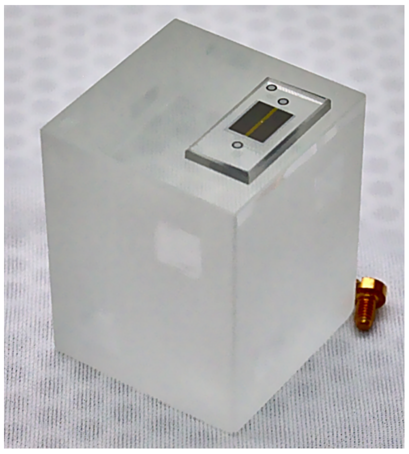


Figure 3.2.: Photograph of the target crystal of a TUM40 detector. The carrier with the evaporated TES in black is seen. The holding clamps, light detector and the detector housing are not mounted. This illustration is taken from [32].

Table 3.1.: Resulting parameters of the pulse shape model given by equation (2.7) fit to the absorber as well as the carrier standard event of TUM40. The pulse shape model is expanded by a line to consider a baseline offset d and a slope k. The fit is plotted in figure 3.3.

Fit Parameter	Absorber Template	Carrier Template
d	0.0020 V	0.0008 V
k	0.000045 V/ms	0.000004 V/ms
t_0	-1.694 ms	-0.902 ms
A_n	1.452 V	1.239 V
A_t	0.352 V	0.049 V
τ_n	19.539 ms	2.110 ms
τ_{in}	2.587 ms	0.114 ms
τ_t	76.255 ms	29.940 ms

contact with the TES contrary to the absorber. This statement is confirmed by the fitting parameters seen in table 3.1.

3.2. Former Discrimination Methods

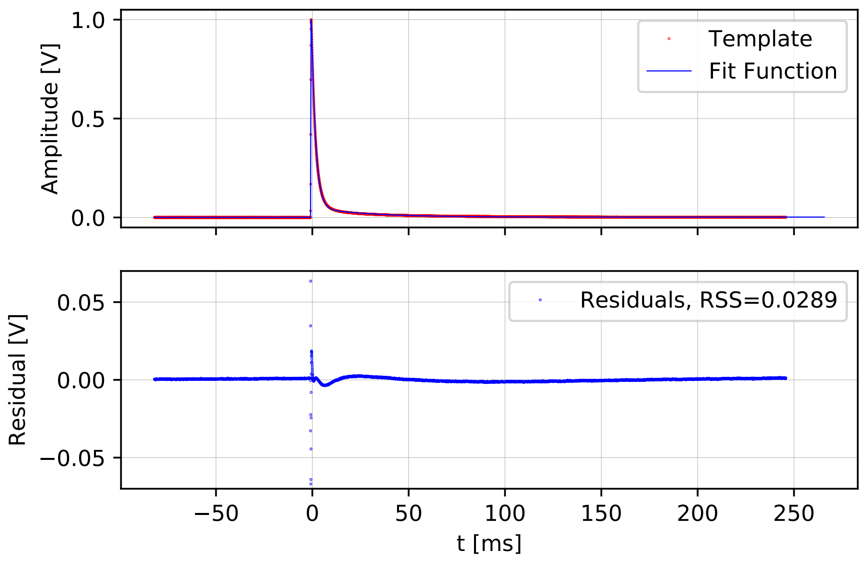
TUM40 pulse shape discrimination methods have been studied in detail in [26]. In the following part the main concepts of the former publication are summarized.

3.2.1. RMS Ratio

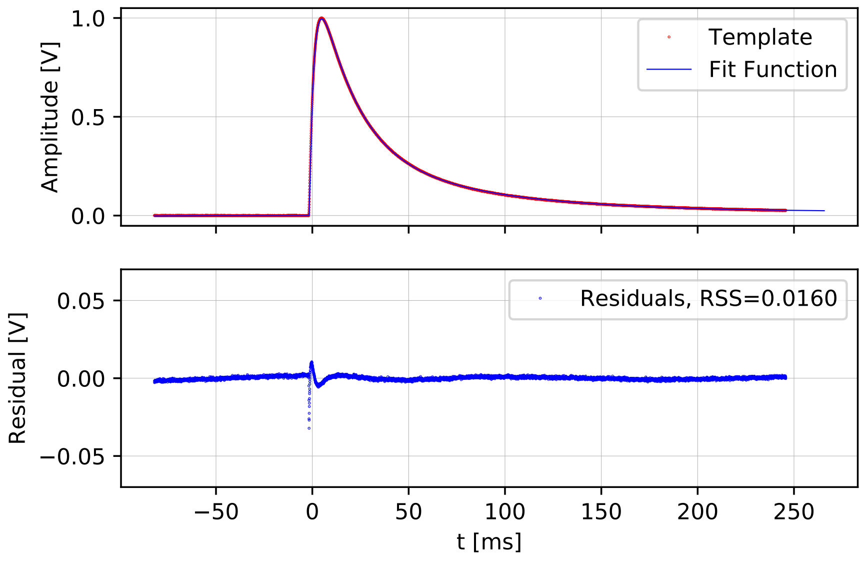
This method is based on the template fit which is discussed in section 2.3.2. To every observed pulse the absorber and the carrier standard event is fit resulting in two sets of fitting parameters as well as two values for the MSE and RMS respectively. The RMS (Root Mean Squared Error) is defined as the square root of the MSE. Normalizing both RMS values leads to the RMS ratio which is defined as

$$\text{RMS ratio} = \frac{\text{RMS}_{CarrierFit} - \text{RMS}_{AbsorberFit}}{\text{RMS}_{CarrierFit} + \text{RMS}_{AbsorberFit}}, \qquad \text{RMS ratio} \in [-1, 1]. \tag{3.1}$$

The RMS ratio seen in equation (3.1) ranges from -1 for a perfect carrier template fit to 1 for a perfect absorber template fit. Consequently absorber events are expected to result in a positive RMS ratio and vice versa for carrier events. The RMS ratio can be plotted as a function of the deposited particle energy which leads to two distinct bands. At low energies (low amplitudes) the noise dominates the RMS leading to an overlap of the two bands, therefore the pulse shape discrimination based on the RMS ratio fails. A more sophisticated method of pulse shape discrimination is discussed as follows.



(a) Equation (2.7) fit to the carrier template.



(b) Equation (2.7) fit to the absorber template.

Figure 3.3.: Depiction of the carrier and absorber templates fit with equation (2.7) based on the basin hopping optimization algorithm. The resulting parameters are summarized in table 3.1.

3.2.2. Neural Network

Neural networks are universal function approximaters and can be used in the context of a classification task. It is proposed that the underlying structure in the data can be learned by a neural network in order to discriminate pulses at very low energies where the RMS ratio fails. The concept of neural networks is discussed in detail in section 5. In this context neural networks work with the following eight input parameters that are provided by the standard event fit and the pulse itself.

Rise Time 10-70 % is directly evaluated on the raw pulse and described in section 2.3.1.

Decay Time is directly evaluated on the raw pulse and described in section 2.3.1.

Absorber Amplitude A_{Absorber} is provided by the standard event fit with the absorber template.

Absorber RMS is provided by the standard event fit with the absorber template.

Carrier Amplitude A_{Carrier} is provided by the standard event fit with the carrier template.

Carrier RMS is provided by the standard event fit with the carrier template.

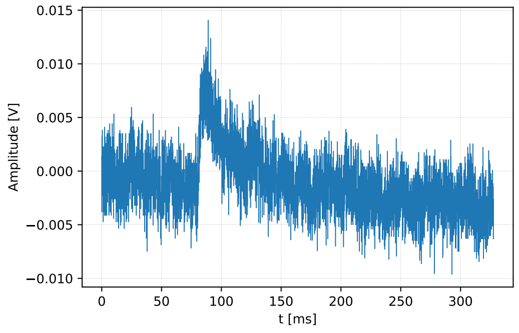
Amplitude Ratio is defined as $A_{\text{Absorber}}/A_{\text{Carrier}}$.

RMS Ratio is defined in equation (3.1).

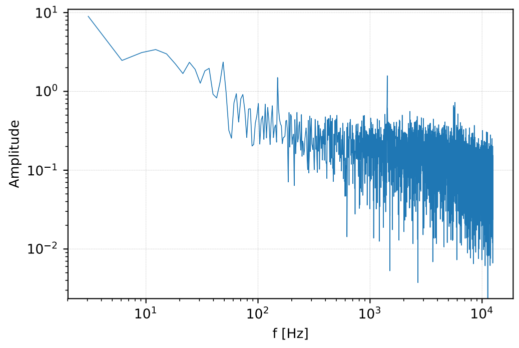
The neural network was trained on data generated with the later introduced method seen in section 4 with two different optimization methods. The favored neural network architecture consists of three linear layers with eight and seven nodes per hidden layer. This discrimination method outperformed the RMS ratio at low energies and was used to analyze real CRESST data as outlined in [26]. Both of the former methods relied on parameters given by the standard event fit. In the following chapter discrimination methods are discussed that work with the raw pulse itself.

3.3. Potential Discrimination Methods

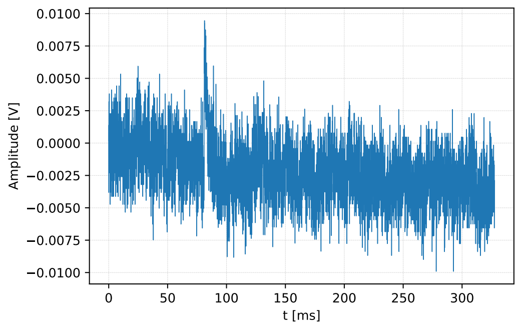
The former investigated discrimination methods are solely based on quantities resulting from the raw data analysis in combination with the standard event fit. In contrast different discrimination methods can be applied on the raw pulse itself or on the Fourier transformed pulse. Figure 3.4 illustrates an absorber as well as a carrier pulse raw and Fourier transformed. As observed the noise becomes the dominant part in a signal at low energies (low amplitudes), this effect is also problematic in the standard event fitting method. Additionally, pulses are likely to rise on different time stamps in the record due to the triggering algorithm. In the next parts discrimination methods in the real as well as in the Fourier space are discussed.



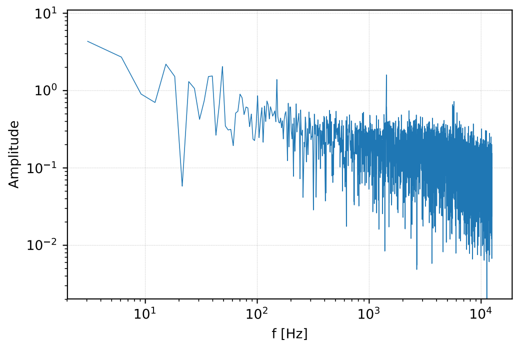
(a) Generated absorber event with and energy of 1 keVee.



(b) Fourier transformation of the generated absorber event.



(c) Generated carrier event with and energy of 1 keVee absorber equivalent.



(d) Fourier transformation of the generated carrier event.

Figure 3.4.: Illustration of generated absorber and carrier events. The energy of the absorber event corresponds to 1 keVee, the same corresponding scaling parameter was used for the carrier event (absorber equivalent). The left column depicts the raw pulse and the right column illustrates the Fourier transformation, note that the amplitude for a frequency of zero is not seen due to the logarithmic scale. The method to generate artificial pulses is described in chapter 4.

3.3.1. Discrimination in Fourier Space

Many machine learning algorithms such as SVM (Support Vector Machines) or neural networks can be used to discriminate pulses in the Fourier space. Different characteristical frequencies are expected for absorber and carrier events respectively as also seen in figure 3.4. An algorithm that is used in CRESST-III to trigger pulses acts in the Fourier space and can potentially be used to discriminate TUM40 data as outlined below.

Optimum Filter

In this chapter the principle of the optimum filter method will bis summarized as well as how this method could be applied to discriminate between the two TUM40 pulse shapes based on [33]. The core idea of the optimum filter method is to apply a filter function on the pulse such that signal-like features are amplified and the noise is removed. Based on the Fourier transformation of a typical signal shape $\mathrm{s}(\omega_\mathrm{k})$ which can be obtained by the absorber standard event and a power spectral density of the stationary noise $\mathrm{N}(\omega_\mathrm{k})$ that can be obtained by measuring noise of the detector as discussed in section 4.2.1 the following transfer function is defined as

$$H(\omega_k) = h\frac{s^*(\omega_k)}{N(\omega_k)}e^{-j\omega_k i_M} \ . \tag{3.2}$$

In equation (3.2) h denotes a normalization constant and $\mathrm{i_M}$ denotes the filter delay. A signal can be filtered by multiplying the Fourier transformed signal with the transfer function[1] leading to an amplification of the frequencies that are signal-like given by $\mathrm{s}(\omega_\mathrm{k})$ and reducing the noise like frequencies as the signal is divided by $\mathrm{N}(\omega_\mathrm{k})$.

In order to discriminate different pulse shapes the standard event used to derive $\mathrm{s}(\omega_\mathrm{k})$ can be filtered to calculate a filtered standard event. This filtered standard event can be fit to any filtered signal by aligning the maximum of the filtered pulse with the maximum of the filtered standard event and scaling the filtered standard event to the amplitude of the filtered signal. Based on the fit a shape indicator is defined as follows

$$SI = \sum_{i=0}^{L-1} \frac{(y_i^f - f_i)^2}{\sigma_L^2(L-2)} \ . \tag{3.3}$$

In equation (3.3) $\mathrm{y_i^f}$ is the filtered signal, $\mathrm{f_i}$ is the fit filtered standard event, σ_L^2 is the expected noise and L is the length of the signal. If the filtered pulse is of the same shape as the filtered standard event the numerator in equation (3.3) would only resemble the present noise on the filtered pulse. As the noise is cancelled by σ_L^2 in the denominator the SI evaluates to approximately one for equal pulse shapes and higher numbers for non matching pulse shapes. If the absorber standard event is used to derive the filtered standard event this former outlined procedure is a potential method to discriminate the TUM40 pulse shapes.

[1]Many additional details need to be considered in order to filter a signal with the transfer function that are outlined in the publication [33].

3.3.2. Discrimination in Real Space

Many machine learning algorithms can be used in order to discriminate the TUM40 pulse shapes. As the former work seen in [26] used neural networks together with the parameters resulting from the standard event fit as described above it is from great interest how neural networks would perform on the raw data, hence skipping the standard event fitting procedure. The remaining part of this work focuses on the TUM40 pulse shape discrimination problem investigated with neural networks on the raw pulse.

4. Artificial Pulse Generation

In chapter 3 the TUM40 detector design with its two distinct pulse shapes was introduced. The resulting classification problem and additionally former as well as future discrimination methods were discussed, concluding that this work focuses on pulse shape discrimination with the raw pulses and neural networks. As this classification task is a supervised learning problem, training data with known class labels are needed. In this chapter a method that generates pulses that resemble the real nature of the observed data is discussed. This artificial pulse generation method with its two steps is sketched in figure 4.1. Detector noise is superimposed with signal-like shapes (templates) of the two distinct pulse classes resulting in an artificial pulse that resembles the nature of a real pulse. The former two ingredients namely the template as well as the noise are discusses as follows.

4.1. Pulse Shape Templates

In chapter 2.3.3 it is outlined that with a known radioactive calibration source two templates are generated, one for the absorber and one for the carrier respectively. Based on the pulse shape discussion in chapter 2.2.2 the two templates were fit with equation (2.7) as summarized in section 3.1.1 resulting in two analytic functions. The templates as well as the fit functions are seen in figure 3.3. In summary the templates and the standard event fit functions are available to generate pulses. The analytic fit functions can be shifted in time freely whereas samples are missing at the beginning/end of the record if the templates are shifted in time. Additionally, the analytic fit functions are noise-free. In contrast the template is an average of a finite number of measured pulses, hence the variance of the noise is not completely averaged out[1]. In order to generate pulses that resemble different deposited particle energies the templates are scaled as follows.

Energy Scaling

Following the theoretical discussion of the pulse shape model given in section 2.2.2 the amplitude scales approximately linearly with the deposited particle energy. Experimentally the energy reconstruction is done with heater pulses as discussed in chapter 2.3.3. In order to generate pulses of different energies the energy reconstruction procedure is reversed based on the TUM40 detector response curve seen in figure 2.8. Starting with the wanted energy E on the right ordinate, assuming that the green curve is approximately a line with an observed slope of k=111.78 keV/V and finally resulting in a scaling

[1]A template would only be noise-free if an infinite amount of measured pulses were averaged.

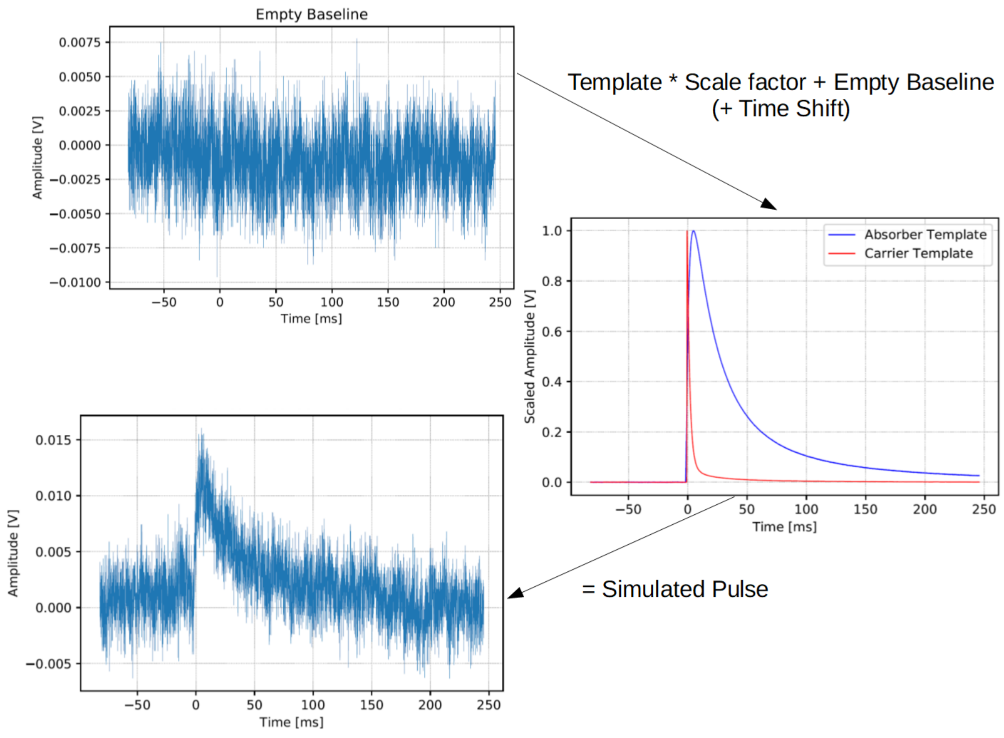


Figure 4.1.: Sketch of the method to generate artificial pulses. Empty baselines are superimposed with scaled and additionally time-shifted templates resulting in generated absorber and carrier events of different energies.

parameter given by s_E=E/k. As a template is scaled to an amplitude of 1 V, multiplying every sample of the template with the scale factor s_E leads to a template that resembles an event originating from a deposited energy of E. This outlined scaling method only provides meaningful results for the absorber template as the calibration is only done for absorber events. However, the same procedure is carried out for the carrier template but in this case the scaled energy E does not resemble the deposited particle energy as the carrier is not calibrated. Furthermore, it is noted that scaling the template, especially to high energies, amplifies the minor present noise component, thus the analytic description is favored to generate artificial pulses.

4.2. Noise

Unavoidable signal that is always present on the detector is denoted as noise. Many sources such as thermal fluctuations, thermal links, electrical resistors etc. produce noise, a detailed investigation is given in [34]. As not every physical aspects is understood, noise simulation methods are not available. Therefore this work solely relies on measurements as follows.

4.2.1. Empty Baselines

Empty baselines are detector readouts where no event is triggered, resulting in many records with 8192 samples. Consequently this measurements should be free from particle interactions and resemble the detector noise characteristics. For TUM40 142591 empty baselines were measured. Several unwanted artifacts, where some are depicted in figure 4.2, are present on the data that were removed with the following cuts.

Derivative Cut The first derivative at every sample is calculated and divided by the RMS resulting from an offset fit in the first n samples. If the maximum sample exceeds the cut limit this record will be removed. This cut is designed to remove empty baselines as seen in figure 4.2a.

Pulse Height Cut The pulse height parameter as introduced in chapter 2.3.1 is calculated. If the pulse height exceeds the cut limit this empty baseline will be removed. This cut removes particle events happening in the time window of the readout as seen in figure 4.2b.

Step Cut Firstly an offset is fit into the first n samples. Secondly the percentage of residuals that are below or above n times the standard deviation based on the offset fit are counted. If the calculated percentages exceed a certain level this empty baseline will be removed. This cut is designed to remove artifacts as seen in figure 4.2c. Those artifacts originate from the fact that the SQUID shows discrete baseline levels, hence a non-zero average baseline level can occur and moreover jumps of the baseline level are likely.

Table 4.1.: Summary of the parameters for the four outlined cuts. From the initial 142591 empty baselines 103178 survived all four cuts.

Cut Type	n_{Samples}	n_σ	Cut Limit	n_{Cut}
Derivative Cut	200	-	5.0	4679
Pulse Height Cut	200	-	0.03 V	2468
Step Cut	200	3	50.0 %	25738
Right Minus Left Baseline Cut	200	5	$5{\cdot}\sigma$	32749

Right Minus Left Baseline Cut An offset is fit in the first and the last n samples. If the fit offset of the last samples lies outside the range given by n times the standard deviation based on the fit of the first samples the empty baseline will be removed. This cut is designed to remove artifacts as seen in figure 4.2d.

Table 4.1 summarizes the applied cuts. From the initial 142591 empty baselines 103178 survived all four cuts. Consequently about 100000 empty baselines are available to generate artificial pulses, this number can be too little for certain applications. A method that is based on the measured empty baselines seen above but provides infinite amounts of simulated noise is discussed as follows.

4.2.2. Baseline Simulation

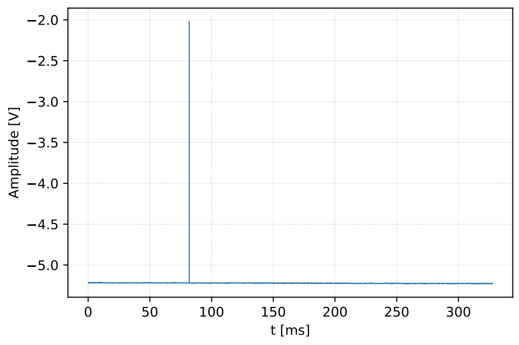
Based on the measured empty baselines a spectral power density is calculated that is used in a sampling method as outlined in [35]. This method provides an infinite amount of empty baselines and its uses for the CRESST experiment will be investigated in the work of [36].

4.3. Generated Datasets

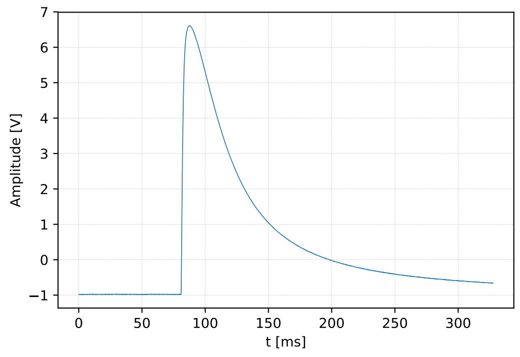
As seen in figure 4.1 an empty baseline is superimposed by a known template which is scaled and maybe additionally shifted in time resulting in an artificially generated pulse. Different available templates and different forms of empty baselines were discussed. Based on the former considerations three different datasets were generated as follows.

4.3.1. Datasets with Measured Noise

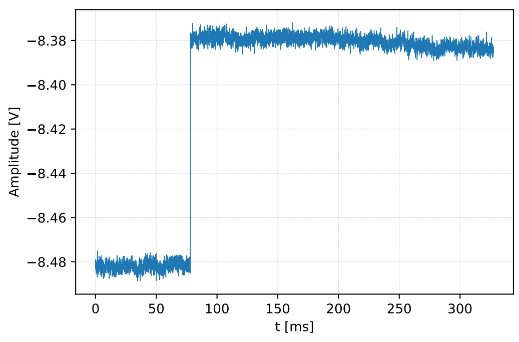
Two datasets are created with the surviving empty baselines by superimposing every empty baseline with one carrier and one absorber event resulting in approximately 200000 generated pulses per dataset. The templates were used for one and the fit templates were used for the other dataset. The only difference is that for the fit templates an additional offset in time was used. This time offset is uniformly sampled from $t_{\text{shift}} \in [-10\text{ ms}, 10\text{ ms}]$ and added to the time offset seen in table 3.1. The energies were uniformly sampled from $E \in \{0.342, 0.4, 0.5, 0.6, 0.7, 0.8, 1.0, 3.0, 7.0, 11.0\}$ keV, this set of energies was also used



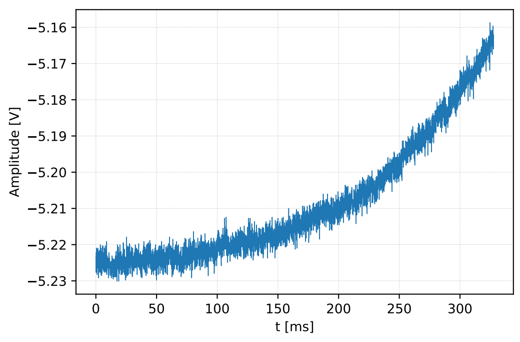
(a) One sample has a very high value.



(b) An event directly happening in the read-out time window.



(c) The SQUID baseline level jumps.



(d) A rise observed in the readout window due to a temperature increase of the detector.

Figure 4.2.: Several examples of artifacts observed in the measured empty baselines of the TUM40 phonon channel. The amplitudes correspond to the SQUID outputs.

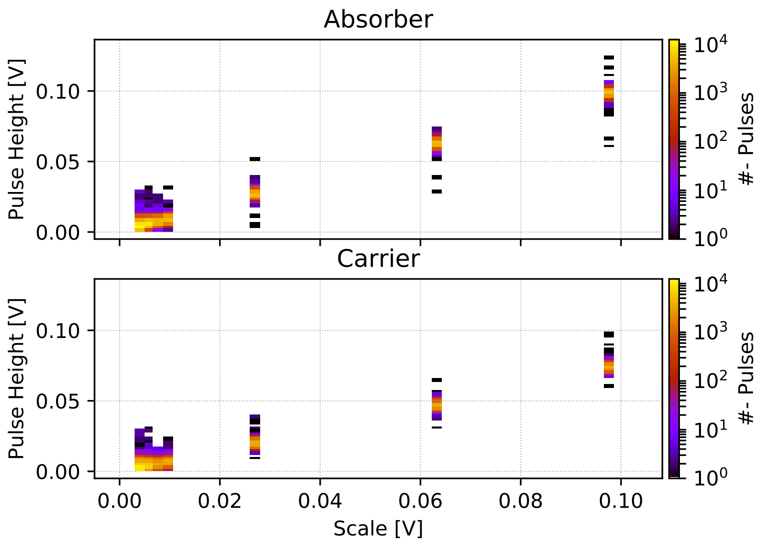


Figure 4.3.: Depiction of the determined pulse height parameter as a function of the scaling parameter for the dataset described in section 4.3.1.

in [26]. In summary two equal datasets were generated that only differ by an additional time shift.

In figure 4.3 the computed pulse heights are plotted as a function of the used template scale factors for the absorber as well as the carrier templates. The distinct scale factor populations given by the discrete values of the former mentioned set of energies is observed. For the generated absorber events the pulse height parameter is approximately the scale factor as expected. Contrary, the pulse height parameter underestimates the scale factor regarding the carrier events. This is due to the fast rise and decay of the carrier pulse shape in combination with the 50 sample average that is used to derive the pulse height, the carrier peak is averaged out and hence the resulting pulse height is lower.

4.3.2. Dataset with Simulated Noise

This dataset is generated by using simulated empty baselines in combination with the templates as outlined in [36] resulting in a dataset without an additional time shift. Figure 4.4 depicts the pulse height parameter as a function of the scale parameter. Much higher energies were generated and the carrier pulse height underestimation is observed in analogy to the former discussed dataset. It must be mentioned that the generated pulse heights do not resemble the real data. This is due to the fact that at higher energies the TES saturates and hence the pulse shape changes.

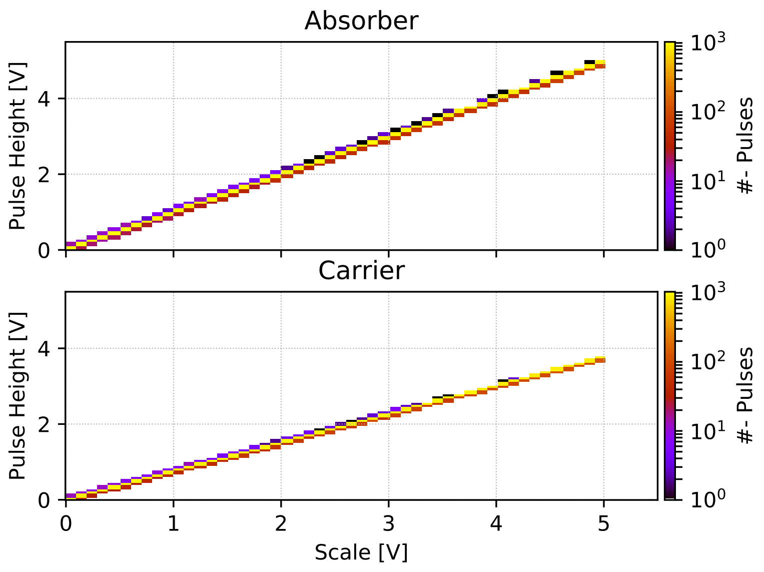


Figure 4.4.: Depiction of the determined pulse height parameter as a function of the scaling parameter for the dataset described in section 4.3.2.

5. Neural Networks

A neural network is a universal function approximation algorithm that is widely used in many different application due to its powerful performance. Ranging from image classification, pattern recognition and speech recognition to complete AI solutions and many more machine learning applications. Machine learning tasks can be divided into the following three main categories.

Supervised Learning In this framework every input is labeled and the machine learning algorithm learns to predict these labels given a new input. If the label is a real number it is referred to as a regression task whereas in a classification task the label is a natural number representing a class. In terms of statistics the algorithm is meant to learn the conditional probability distribution p(y|x) where y is the label and x is the input.

Unsupervised Learning In this framework labels are not provided with the data. The algorithm is expected to learn the underlying structure of the data. Examples are clustering algorithms where similar datapoints are combined in different classes, or learning the conditional distribution p(x|y) which means that the algorithm generates data x of the form y.

Reinforcement Learning In reinforcement learning the algorithm is expected to make certain decisions to get to a desired end result. Learning is done based on the outcome resulting that results from a combination of all former decisions. A common example is the game chess. The algorithm can decide its move on every round and learns on the final outcome of the game which is win or loss.

Neural networks are used with great success in all three former mentioned machine learning ares. This work focuses on the TUM40 classification task where the input is a pulse (a vector of 8192 dimensions) and the label is zero or one for carrier and absorber event, respectively. In this chapters first part the basic concept of a neural network is outlined, in its second part a complete walk through a supervised learning problem will be given and in its final part a selection of state of the art neural network techniques are discussed.

5.1. Fully Connected Feed Forward Neural Network

Closely following the discussion seen in [37] neural networks realize the idea of a very simplified brain model. In this simplified model the brain is a composition of neurons that process information and synapses that connect the neurons in order to pass the information from one neuron to the others. The strength of this connection is learnable and determines how input information is processed. This former concept is expressed in mathematical terms as follows.

$$o_j = \sigma\left(\sum_{i=1}^{N} w_{ij}x_i + b_{0j}\right), \qquad 1 \leq j \leq M, \qquad \sigma(x) = \frac{1}{1+e^{-x}}\ . \tag{5.1}$$

In equation (5.1) $\mathrm{x_i}$ are the inputs multiplied by the learnable weights $\mathrm{w_{ij}}$ and a learnable bias[1] $\mathrm{b_{0j}}$ that represent the synapses. This calculated linear combination of the inputs with the learnable weights is inserted into an activation function σ which is resulting in the output of one neuron $\mathrm{o_j}$, a neuron is also referred to as node. The activation function σ is a so-called Sigmoid function and is the historical first used activation function, many additional activation functions have been found as seen in section 5.3.1.

M sets of weights that lead to M outputs form a layer where all N inputs are connected to all M outputs, hence it is a fully connected layer with N·M+M learnable weights. Fully connected layers can be stacked, this is done by taking the outputs of one layer as inputs to the next layer. Many stacked layers are referred to as a deep neural network or deep learning. It is noted that the information of the network is only fed forward from the first layer over the intermediate hidden layer(s) to the last output layer. In summary many layers represented by equation 5.1 can be stacked together in order to form a fully connected deep neural network (FNN) that is also illustrated in figure 5.1. In contrast many other forms of architectures are imaginable, such as connections between every node, self-connections etc. which is discussed in section 5.3.5. But this simple model given by equation (5.1) can approximate any continuous function on compact subsets of $\mathrm{R^n}$ with a finite number of weights [38]. However, it is unclear how to receive the corresponding weights. This is a major challenge and will be examined in chapter 5.2. Additionally, one layer can in principle approximate any function as mentioned above but deep neural network architectures seem to perform superior. Again it is not clear how many layers are best or the number of nodes per layer etc., this lack of the a-priori knowledge of such hyperparameters is denoted as black magic of deep learning, hyperparameter search methods are discussed in section 5.3.4. In conclusion a summary of many used neural network terms is given as follows.

Node, Neuron are the composition of the linear combination of its inputs with the weights and the applied non-linearity, as seen as blue rings in figure 5.1. A layer typically consists of many nodes.

Parameters are the learnable weights as well as bias of the neural network.

Hyperparameter denotes every parameter which is a-priori unknown but crucially determines the performance of a neural network. For example the number of layers, number of nodes per layer, batch size, size of kernel etc.

[1]Usually the bias is included into the set of learnable weights as $\mathrm{w_{0j}}$, hence the input is extended with $\mathrm{x_0}$ that is always set to one.

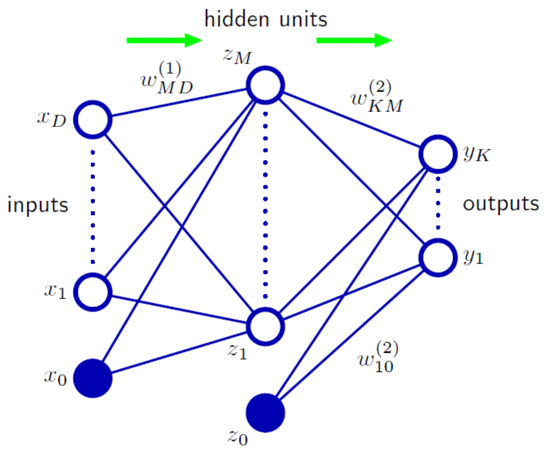


Figure 5.1.: Sketch of a two layer feed forward neural network. The blue circles represent the network nodes. The two inputs are fully connected to the next layer over the respective weights. In the center two nodes represent the hidden nodes and are fully connected to the two nodes of the output layer. The blue filled circles are the inputs always equal to one representing the bias terms. The above model has four weights and one bias per layer equaling to an overall number of twelve learnable parameters. The green arrows show the forward information flow. This illustration is taken from [37].

Layers The definition of a layer in a neural network is not uniform. One definition is to count the different compositions of hidden nodes as hidden layer where the input and output nodes are not respected. Another definition is that every set of learnable weights is counted as a layer. The network depicted in figure 5.1 has one (hidden) layer based on the former and two layers based on the latter definition.

Sample, Feature A sample is one single element x that can be the input of a neural network, a sample is usually a vector. A feature is one item x_i of the input vector.

Batch, Mini-Batch, Iteration, Epoch In a supervised learning algorithm labeled data are used in order to learn the parameters. A training batch is referred to the whole data that are used for training. Training a neural network is done on subsets of the batch one by one, where every subset is denoted as mini batch. An iteration is the training procedure only on one mini batch, and if all mini batches that form the training batch are used once to train a neural network it is denoted as epoch. Neural networks are usually trained for many epochs.

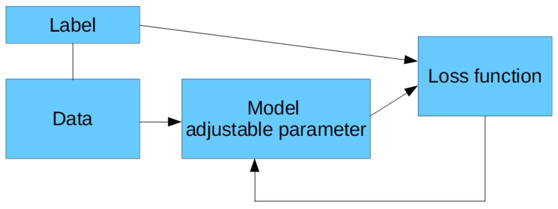


Figure 5.2.: Depiction of a supervised learning task. Predictions of the model based on the given data are compared to the the true label of the data. The deviation of the former two quantities measured by the loss function is used to adjust the parameters of the model in order to learn the labels of the data.

5.2. Supervised Training Cycle

The aim of this chapter is to outline the commonly used method of adjusting the weights of a neural network to perfectly fit the given data. Many of these steps are not unique to neural networks but are used in the field of supervised machine learning in general. Figure 5.2 depicts the different steps of a supervised learning task. Data are fed through the neural network to predict the respective label, the loss function measures the deviation between the prediction and the label of the given data. Lastly, the weights are adjusted to minimize this deviation. In the context of neural networks this training cycle is repeated for many epochs. The first part of this chapter concentrates on the statistical framework to derive a loss function commonly used in classification and regression. Secondly, details of the data are discussed. In the third part a method to evaluate the gradient of the loss function with respect to the neural network weights is introduced. And lastly, ways of using the gradient information to adjust the learnable weights are outlined.

5.2.1. Loss Function

To derive a loss function an underlying statistical model has to be defined by considering the conditional distribution of the targets given the data. Based on this conditional distribution the neural network models the conditional mean E(t|**x**,**w**) which is a function of the data vector **x** and the learnable parameters **w**. Given a two-class classification problem the underlying process is the following Bernoulli distribution

$$f(t|\boldsymbol{x},\boldsymbol{w}) = p(\boldsymbol{x},\boldsymbol{w})^t\,(1-p(\boldsymbol{x},\boldsymbol{w}))^{1-t}\,, \qquad E(t|\boldsymbol{x},\boldsymbol{w}) = p(\boldsymbol{x},\boldsymbol{w}) \tag{5.2}$$

In equation (5.2) t is zero or one representing the two classes and p(**x**,**w**) denotes the probability of a sample **x** belonging to class t. As p(**x**,**w**) is the conditional mean this quantity is modeled by the neural network that gets a sample **x** as an input and predicts the class probability (a number between zero and one) as an output. In order to derive the parameters of a statistical model given some data the parameters are adjusted such that the probability of observing the given data is maximized. This procedure is referred to as maximum likelihood. Assuming independent data the probability of obtaining N samples representing the data is given by the likelihood function as follows

$$\mathcal{L} = \prod_{i=1}^{N} f(t_i|\boldsymbol{x}_i, \boldsymbol{w}) \ . \tag{5.3}$$

To derive the maximum likelihood solution for the parameters equation (5.3) is usually transformed by taking the logarithm[2] and switching the sign which leads to the loss function. Due to switching the sign the loss function has to be minimized. For the two-class classification problem the loss function is derived as

$$L(\boldsymbol{w}) = -\sum_{i=1}^{N} \left(t_i \ln p(\boldsymbol{x}_i, \boldsymbol{w}) + (1 - t_i)\ln(1 - p(\boldsymbol{x}_i, \boldsymbol{w}))\right) \ . \tag{5.4}$$

Equation (5.4) is referred to as the **binary cross-entropy loss** function. A K class classification problem can be derived similarly based on the multinomial distribution given by

$$f(t_1, \cdots, t_K|\boldsymbol{x}) \propto \prod_{i=1}^{K} p_i(\boldsymbol{x}, \boldsymbol{w})^{t_i}, \qquad \sum_{i=1}^{K} p_i = 1, \qquad t_i \in \{0, 1\}, \qquad \sum_{i=1}^{K} t_i = 1 \tag{5.5}$$

$$L(\boldsymbol{w}) = -\sum_{i=1}^{N}\sum_{j=1}^{K} t_{ij} \ln p_j(\boldsymbol{x}_i, \boldsymbol{w}) \ . \tag{5.6}$$

In that case the neural network has K output nodes representing the probabilities of x belonging to the different K classes, the sum of all output nodes equals to one. Equation (5.6) is labeled as **cross-entropy loss**. Regression problems can be modeled via a normal distribution leading to the following loss function based on the same derivation outlined above leading to

$$L(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N} \left(y(\boldsymbol{x}_i, \boldsymbol{w}) - t_i\right)^2 \ . \tag{5.7}$$

Equation (5.7) is denoted as the **MSE** (Mean Squared Error). Depending on the specific problem a corresponding loss function can be derived as shown above. The weights are determined by minimizing the loss function respectively.

[2]This transformation is valid as the logarithm is a monotonous function hence the maximum/minimum remains at the same argument of the function. For independent data the likelihood function is a product of the density of each sample, based on the calculation rules of logarithms this product transforms in a sum and the solution is attained easier compared to the raw likelihood function.

5.2.2. Data Preparation

In the data preparation step the raw data are adapted to suit the machine learning algorithm. Many steps are needed depending on the given data. The following two steps are of great importance.

Data Pre-Processing

Generally the input features $\mathrm{x_i}$ differ in their range, thus the mean as well as the standard deviation are not equal across all features. As the input of a node is the linear combination of the input features with the weights, input features of different scales (different standard deviations) lead to different relative importance of the input features assuming that the weights are of equal magnitude. To avoid this issue every input feature is transformed to a mean of zero and a standard deviation of one as follows

$$\tilde{x}^{(i)} = \frac{x^{(i)} - \bar{\mu}^{(i)}}{\bar{\sigma}^{(i)}} \ . \tag{5.8}$$

In equation (5.8) $\bar{\mu}^{(i)}$ is the estimator of the mean and $\bar{\sigma}^{(i)}$ is the estimator of the standard deviation per feature. If the features are approximately of the same scale the standard deviation may be approximated over all features. Many more ways of pre-processing are possible such as min-max scaling or using the median instead of the mean, however the particular form of data pre-processing strongly depends on the data structure. The procedure given by equation (5.8) is sufficient for many applications.

Data Set Splits

For a supervised learning task the parameters of a model are determined by minimizing the loss function with respect to the parameters for given data. Usually, the data are split into three distinct subsets. One is used for training (training set), the other one is used to evaluate the model at different training steps (evaluation set), if the training is finished and the best possible evaluation loss is archived the model is tested on the third dataset (test set). This partitioning serves many purposes. In iteration based training it can be measured if the model overfits the data as a function of the training step which is discussed in detail in section 5.3.2. Furthermore, the performance of different network architectures (hyperparameter configurations) can be compared based on the performance of the model on the evaluation set. For the latter purpose this outlined dataset split method is a special case of cross-validation.

5.2.3. Backpropagation Algorithm

The optimal parameters of a neural network are found by minimizing the loss function given the training set. As the loss function and the neural network is nonlinear deriving the gradient and setting it to zero will lead to a set of nonlinear equations which is impossible to solve analytically. Therefore, another common way of finding a minimum is to initialize the parameters randomly and refine them in an iterative fashion. This is done by evaluating the gradient of the loss function with respect to the weights and then updating the weights by stepping into the negative gradient direction which should decrease the loss function. This procedure is referred to as gradient decent. Additionally, many algorithms exist that define the exact way of updating the weights with the gradient information which are discussed in chapter 5.2.4. This chapter will focus on deriving the gradient based on the chain rule of differentiation for a fully connected feed forward network summarizing the outline seen in [37]. The generic implementation of computing the gradient for many different neural network architectures is very complicated and is discussed in detail in [39]. In analogy to equation (5.1) the forward pass of one layer of a fully connected feed forward neural network can be written as

$$o_j = h(a_j), \qquad a_j = \sum_i w_{ji} o_i \ . \tag{5.9}$$

In equation 5.9, $\mathrm{o_j}$ denotes the outputs of a layer which may be the inputs of the next layer and h is the applied activation function. Based on the discussion above the loss function usually decomposes into a sum of the training samples as follows

$$L(\boldsymbol{w}) = E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w}) \ . \tag{5.10}$$

Based on equation (5.10) the gradient over all n samples is given by the sum of the individual gradients given by

$$\frac{\partial E}{\partial w_{ji}} = \sum_{n=1}^{N} \frac{\partial E_n}{\partial w_{ji}} \ . \tag{5.11}$$

The derivation of the gradient for the individual parts of the sum in the loss function seen in equation (5.11) is evaluated with the chain rule off differentiation as follows

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \delta_j o_i, \qquad \delta_j = \frac{\partial E_n}{\partial a_j}, \qquad \frac{\partial a_j}{\partial w_{ji}} = o_i \ . \tag{5.12}$$

The third part in equation (5.12) is simply the derivative of equation (5.9) with respect to $\mathrm{w_{ji}}$. The second part depends on the examined layer in the network. For the output layer the quantity δ_k evaluates as

$$\delta_k = h'(a_k)\frac{\partial E_n}{\partial o_k} \ . \tag{5.13}$$

$\mathrm{E_n}$ of equation (5.13) is the loss function and depends on the form of the problem as discussed in chapter 5.2. For the hidden layers the effect of lower layers on the loss function will be taken into consideration by applying the chain rule as well as the product rule of differentiation given by

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j} \ . \tag{5.14}$$

The last term in equation (5.14) denotes the change of the outputs of a lower layer with respect to the outputs of the above layer and can be derived from equation (5.9) given by

$$\frac{\partial a_k}{\partial a_j} = h'(a_j) w_{kj} \tag{5.15}$$

and with equation (5.14) and (5.15) leading to the final expression for δ of hidden layers given by

$$\delta_j = h'(a_j) \sum_k \delta_k w_{kj} \ . \tag{5.16}$$

In summary equation (5.9) denotes the forward pass of an input through a fully connected feed forward neural network. After the forward pass the gradient of all weights is derived by equations (5.11) and (5.12) with δ that is given by equation (5.13) for the output layer and equation (5.16) for the hidden layers. The former method is called back-propagation because δ can only be evaluated starting from the output layer leading back to the input layer, hence the gradient information flows backwards. Given the gradient information the parameters can be updated as follows.

5.2.4. Parameter Update Methods

The former chapter described how the gradient of the weights can be derived in an efficient way. In this chapter many ways of updating the weights based on the gradient information will be discussed. The first simple approach is to update the weights directly with the scaled gradient given by

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \lambda \nabla E(\boldsymbol{w}_t), \qquad \lambda > 0 \ . \tag{5.17}$$

The weight update rule seen in equation (5.17) is referred to as gradient decent. λ is the learning rate and defines how far the step will be taken in the direction of the gradient, t denotes the number of iteration. The learning rate λ is a hyperparameter, consequently the appropriate value is a-priori unknown and needs to be found by hyperparameter search algorithms outlined in section 5.3.4. Based on equation (5.17) a momentum term can be introduced and is given by

$$\boldsymbol{v}_{t+1} = \epsilon \boldsymbol{v}_t - \lambda \nabla E(\boldsymbol{w}_t), \qquad 0 \leq \epsilon \leq 1, \qquad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{v}_{t+1} \ . \tag{5.18}$$

$\mathbf{v}$ in equation (5.18) is a running average of the past gradient information and can be interpreted as follows. The loss surface depending on the weights can be viewed as a potential and the momentum term as the velocity of a ball running downhill. A change in direction of the weight vector will not be instant because the momentum term somewhat keeps the former direction based on the parameter ϵ. At t+1 $\mathbf{v}_t$ is given prior to the gradient evaluation, hence a update on the weights can be made before the gradient is computed. This method is called NAG (Nesterov's Accelerated Gradient) and is given by

$$\boldsymbol{v}_{t+1} = \epsilon \boldsymbol{v}_t - \lambda \nabla E(\boldsymbol{w}_t + \epsilon \boldsymbol{v}_t), \qquad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{v}_{t+1} \ . \tag{5.19}$$

It can be shown that equation (5.19) increases learning speed compared to the usual momentum term as seen in [40], additionally, a detailed discussion about momentum based methods is given in that publication. The above methods keep a constant learning rate throughout the whole training procedure. Adam is a method that uses adaptive per parameter learning rates with the following update rules

$$\boldsymbol{g}_{t+1} = \nabla E(\boldsymbol{w}_t), \qquad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \hat{\boldsymbol{m}}_{t+1}/(\sqrt{\hat{\boldsymbol{v}}_{t+1}} + \epsilon) \tag{5.20}$$

$$\boldsymbol{m}_{t+1} = \beta_1 \boldsymbol{m}_t + (1-\beta_1)\boldsymbol{g}_{t+1}, \qquad \hat{\boldsymbol{m}}_{t+1} = \boldsymbol{m}_{t+1}/(1-\beta_1^{(t+1)}) \tag{5.21}$$

$$\boldsymbol{v}_{t+1} = \beta_2 \boldsymbol{v}_t + (1-\beta_2)\boldsymbol{g}_{t+1} \odot \boldsymbol{g}_{t+1}, \qquad \hat{\boldsymbol{v}}_{t+1} = \boldsymbol{v}_{t+1}/(1-\beta_2^{(t+1)}) \ . \tag{5.22}$$

In equations (5.20), (5.21) and (5.22) $\odot$ denotes the elementwise multiplication of vectors, t denotes the iteration steps and is an integer starting from zero. The proposed parameters are α=0.001, β_1=0.9, β_2=0.999 and ϵ=10^{-8}. The ADAM optimizer is a state of the art stochastic optimization algorithm as in detail described in [41].

5.3. Neural Network Modern Practice

In the former chapters the concept of a feed forward neural network was introduced leading to an illustration of the main steps to derive suitable model parameters in a supervised learning framework. As many techniques have evolved in the field of neural networks this chapter will give insights of advances specifically used in the present work: Namely, different activation functions, methods to prevent overfitting, methods that enhance the learning speed, neural network architectures that can deal with different forms of data and methods to search for the best possible configuration of hyperparameters.

5.3.1. Activation Functions

From an historical perspective the first investigated activation function was the Sigmoid as seen in equation (5.1). Many additional functions were introduced in the last decades, the prominent used ones are depicted in figure 5.3 and are summarized based on [42] as follows.

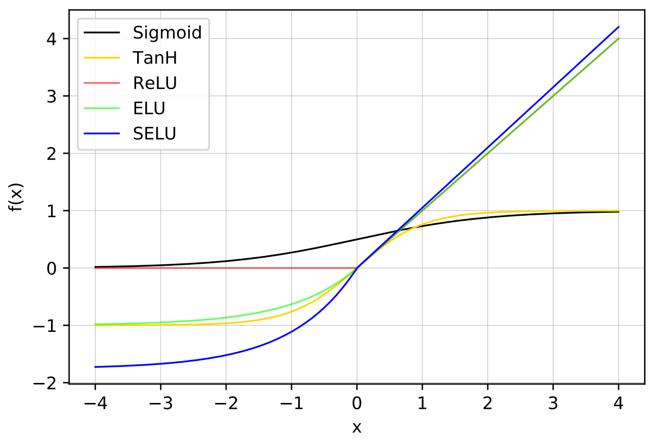


Figure 5.3.: Depiction of different activation functions.

Sigmoid $\sigma(x) = 1/(1+e^{-x})$ The Sigmoid function was the first used activation function and investigated in [38]. Finding suitable network parameters is done by firstly initializing the weights randomly and then refining the weights with backpropagation and certain update rules as described before. The random initialization or strong updates can lead to high/low linear combinations as an argument of the activation function. Considering the Sigmoid function the former issue will lead into a range of the Sigmoid function where the slope is nearly zero. Hence the neuron saturates and the gradient will nearly vanish as it is calculated by a product seen in equation (5.12). This circumstance is resulting in very small updates, hence training is slow. Moreover, this function maps its inputs into the interval $[0, 1]$ which leads to a non zero-centered distribution of the outputs. Due to its drawbacks this function is only used in a two-class classification problem with a single output neuron. The output is interpreted as a probability of the input belonging to a certain class.

Tangens Hyperbolicus $\tanh(x) = 2\sigma(2x)-1$ The Tangens Hyperbolicus shows the same problem with saturation as the Sigmoid function because it is just a re-centered Sigmoid function. However, the distribution of the outputs is now zero-centered.

ReLU $\mathrm{ReLU}(x) = max(0, x)$ The ReLU (Rectified Linear Unit) function does not show the problem with a saturated gradient as the former two activation functions. It is proposed that the learning speed strongly increases as described in [43]. In addition, this function has a very simple gradient expression which is the input itself or zero depending on the sign of the input. As the gradient of the ReLU function is zero for a negative input, the whole gradient of the loss function will be zero based on the backpropagation algorithm seen in equation (5.12). The gradient of a mini batch is the sum of the gradients of all individual samples given by equation (5.11). It will be non-zero if at least the activation of one sample is non-zero. Problems arise if the update of the weights leads into a range where the inputs of the ReLU are negative for every sample in the whole training set. Then the gradient will always be zero and the weight updates are always zero as well, hence this problem is referred to as dying unit.

ELU ELU$(x) = max(0, x) + min(0, \alpha(e^x - 1))$ The ELU (Exponential Linear Unit) does not show the problems of saturated gradients and dying units. In addition, this activation function shifts the average activation towards zero and increases learning speed as described in detail in [44].

SELU SELU$(x) = \lambda(max(0, x) + min(0, \alpha(e^x - 1)))$ The SELU (Scaled Exponential Linear Unit) is an improved version of the ELU. The proposed parameters are α=1.6732 and λ=1.0507. This activation function is examined in section 5.3.3 and is introduced in [45].

Usually a ReLU activation function is used in conjunction with batch normalization for every node of the network expect for the outputs. In a regression task the output activation function is the identity, in two-class classification tasks the output node is a Sigmoid and in a K class classification task the K outputs are normalized to one with the log softmax function.

5.3.2. Methods for Regularization

The aim of a machine learning algorithm is to generalize features from the training data and precisely predict the target of new, unseen data. Overfitting, however, is the opposite as the model fits the parameters in order to remember the training data resulting in a poor performance on new, unseen data. Depending on the number of parameters and on the number of samples in the training data the machine learning algorithm is likely to overfit. An intuitive example would be fitting n datapoints with a polynomial of order n+1. The polynomial would meet every data point of the training data exactly but the fit is meaningless because interpolation would not resemble the underlying structure of the data due to the strong oscillations, hence overfitting occurred. In a statistical context overfitting results from the variance of the model given the training data and competes with the bias, both quantities define the bias variance tradeoff. A high variance in the model connotes that by changing one training sample the model changes significantly. Many ways of preventing the network from overfitting are proposed and often used in combination. The following three methods are standard approaches.

Norm Penalty

The norm penalty is an additional term added to the loss function penalizing the norm of the weights and is given by

$$E(\boldsymbol{w}) = L(\boldsymbol{w}) + \lambda \sum_{w \in W} |w|^q \ . \tag{5.23}$$

In equation (5.23) W denotes all the adjustable parameters present in the model, q is the power of the penalty, λ is the strength of the weight decay treated as a hyperparameter and L is the loss function given by the specific problem as discussed in section 5.2.1. The above regularization is referred to as weight decay or in the case of q=2 L2 penalty. Formally the L2 weight decay term can be derived by Bayesian statistic with the use of a normal prior. A detailed description of norm penalty is given in [37]. Equation (5.23) can be reformulated to an optimization problem restricted by the following condition

$$\sum_{w \in W} |w|^q \leq \eta \ . \tag{5.24}$$

Equation (5.24) restricts the shape and the volume of the solution in the space of weights by η. In analogy to polynomial fitting, weight decay regularizes by tolerating a very high number of free parameters but restricting the value of them. The same procedure is applied in linear models namely the ridge or lasso regression. As the number of parameters of a neural network increases with the number of layers as well as the number of nodes per layer and additionally it is not known how many parameters work best, weight decay is a potential method to prevent overfitting. Usually a L2 penalty is used with the same λ for the whole set of weights.

Dropout

Dropout is a powerful tool to prevent overfitting. It is proposed in [46] and will be summarized based on this publication. The core idea is that the mean output of many uncorrelated neural networks improves the performance significantly by averaging out the variance present in the outputs of many models, this is referred to as bagging. However, it is numerically not feasible to train many networks. Therefore dropout uses one neural network and sets many nodes at random to zero as seen in figure 5.4. By doing so a sub-network is sampled from the 2^{n} possible combinations considering the whole network where n is the number of all nodes. This sampled sub-network is trained with the backpropagation algorithm as described in the former chapter for the current iteration. To evaluate new unseen data in principle the outputs of all sub-networks need to be evaluated and averaged. As this is numerically impossible the whole network is used to evaluate new data but with scaled weights p$\mathbf{w}$. The parameter p is the probability to set a node to zero while training. The former scaling ensures that the expected output of every nodes is the same during training as well as during evaluation[3]. In [46] a dropout rate of 50% is proposed with nearly 0% at the inputs, however the dropout rate can also be treated as a hyperparameter.

Early Stopping

A way of measuring overfitting is implemented by splitting the data into three sets distinct as mentioned in section 5.2.2. In the context of a supervised learning task with

[3]At training time the output of a node is multiplied by zero or one which is sampled from a Bernoulli distribution leading to an average output of p·o. As the sampling is omitted at the evaluation a multiplication of the weights with p results in the same average output as at training time. Another form of scaling can be done during training. To ensure the same mean the weights will be divided by p at training time and no transformation will be applied at evaluation time.

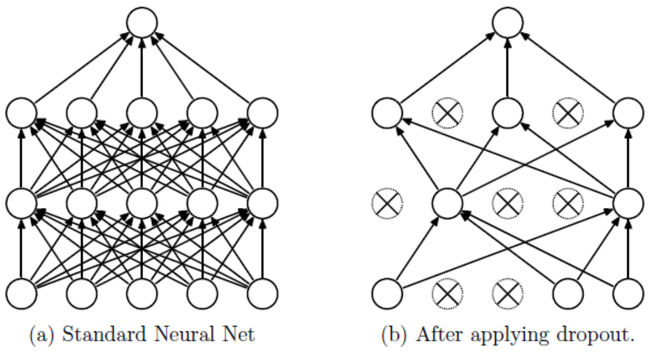


Figure 5.4.: Schematic of the dropout method. (a) shows the whole neural network and (b) shows a sub-network where random nodes are set do zero with a probability of p. This illustration is taken from [46].

iteration based learning the training set is used to train the model by adjusting the weights as described in section 5.2 for neural networks. The evaluation set is used to evaluate the value of the loss function at every epoch. As training is finished and the best possible evaluation loss is archived the model can be tested on the third dataset. Based on this dataset splits the evaluation loss as well as the training loss are monitored as a function of the epoch. The training loss should always decrease because the minimization is done with the training data. However, the evaluation loss will initially decrease but at some point it will increase again due to the algorithm starting to overfit to the training data. Usually the model parameter will be taken where the evaluation loss is minimal, thus further training will not improve the evaluation set loss. In summary this procedure ensures that overfitting of the model is avoided by stopping training early. The decrease and increase of the evaluation loss is usually non-monotonic therefore a minimum in the evaluation loss is likely to be just locally. Based on this consideration early stopping criteria are formulated in [47] and summarized as follows. At every training epoch denoted by (i) the validation loss will be computed and saved. The relative increase of the validation loss relative to the best validation loss up to epoch (i) is defined as

$$GL^{(i)} = 100\left(\frac{E^{(i)}}{\min_{\forall j\in I} E^{(j)}} - 1\right), \qquad I = \{1,\dots,i\}\ . \tag{5.25}$$

Training is stopped if GL of equation (5.25) exceeds a certain percentage and the model with the minimum evaluation error is retained. It is often observed that the evaluation error jumps rapidly in the first training epochs which could lead to a very early stopping based on the former criterion. An improved version should stop training if the validation error rises slowly. This is achieved with the following definition

$$P_k^{(i)} = 1000\left(\frac{\text{mean}_{\forall j\in I} E^{(j)}}{\min_{\forall j\in I} E^{(j)}} - 1\right), \qquad I = \{i-k+1,\dots,i\}\ . \tag{5.26}$$

The quantity given by equation (5.26) measures how much the minimum evaluation error deviates relative to the mean evaluation error in a running window of size k. This quantity can be evaluated after epoch k and will be high if the evaluation error jumps and will be low if the evaluation error smoothly rises/falls with a low slope. Based on the above two quantities the following early stopping criterion can be defined as

$$PQ = \frac{GL^{(i)}}{P_k^{(i)}} \ . \tag{5.27}$$

Training will be stopped if PQ exceeds a certain limit, the model with the minimal evaluation loss will be saved. Equation (5.27) can be interpreted as the global rise of the validation error given by equation (5.25) weighted by the validation error oscillation measured with equation (5.26). Resulting in a criteria which is likely to stop if the validation error slowly rises at a level above the minimum validation error and additionally preventing the algorithm to stop in the first few training epochs. This work uses the definition of equation (5.27) with k=5 and and a limit of PQ of 0.5. It is mentioned that none of the above methods definitely stops training, hence a maximum epoch where training terminates has to be defined.

Methods such as exponential smoothing or running averages are another form of commonly used early stopping algorithms. In summary it is important to use the model with the minimum observed evaluation loss independent of the used early stopping criterion in order to prevent overfitting. The only aim of the early stopping algorithm is to train long enough to reach the best evaluation loss and furthermore prevent from training far beyond the perfect epoch.

5.3.3. Enhancing Learning Speed

As seen in chapter 5.2 the optimization of the weights of neural networks is an ambitious task. Some activation functions saturate or kill the gradient which slows down training immensely. Furthermore, a neural network is a combination of distinct layers, therefore every layer's input distribution changes as the weights are updated at every iteration. This is referred to as covariate shift and slows down training as well. Both problems can be avoided if the outputs of the nodes are somehow normalized to a mean of zero and a standard deviation of one. The following two methods do this in different ways, namely batch normalization introduces a new normalization layer and the self normalizing neural network introduces a new activation function that has the ability to normalize its respective inputs.

Batch Normalization

Batch normalization was introduced by Google and is designed to prevent covariate shift during training and thus increases learning speed. This chapter will summarize the main concept based on the original publication seen in [48]. Batch normalization addresses the covariate shift issue by standardizing the inputs of every layer to ensure a mean of zero and a standard deviation of one. This can be done by computing the covariance matrix at every iteration step and fully standardizing the inputs of every layer. However, as this is computationally very expensive batch norm uses two simplifications as follows.

- The features a treated independently. Hence the standard deviation and the average will be computed per node. Which prevents the calculation of the inverse covariance matrix.
- The average and the standard deviation are not computed over all training sample at every iteration. Rather the mini batch is used to estimate the former quantities.

For every node two new parameters are introduced and the batch normalization of a node (k) writes as follows

$$BN_{\beta,\gamma}(x^{(k)}) = \gamma^{(k)} \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}] + \epsilon}} + \beta^{(k)} \ . \tag{5.28}$$

In equation (5.28) BN is the standardized feature, γ and β are the new learnable parameters, ϵ is a small parameter for numerical stability and the expectations and variances are computed per feature over the mini batch. The former operation is fully differentiable. Batch normalization keeps track of all variances and averages by computing a running exponential average which is used when the model is evaluated. The paper proposes to use the batch normalization layer before the activation function, thus saturated gradients with the Sigmoid function and killing gradients with the ReLU can be avoided. For convolutional neural networks the normalization will be computed over the whole feature map instead of the single nodes.

Self Normalizing Neural Network

A SNN (Self Normalizing Neural Network) maps the mean and variance of one layer's activations to the next layer's activations while the mapped means and variances remain in an interval and additionally get drawn to a fixed point, the detailed definition as well as a detailed discussion is given in [45]. This fixed point is depending on the following requirements on the weights of the respective layer

$$\omega = \sum_{i=1}^{n} \omega_i, \qquad \tau = \sum_{i=1}^{n} \omega_i^2 \ . \tag{5.29}$$

In equation (5.29) n denotes the dimension of the layer's input vector. Whereas batch normalization introduces a new operation after every layer the proposed SNN achieves this self normalizing property with the SELU (Scaled Exponential Linear Unit) activation function in conjunction with a weight initialization that satisfies the conditions seen in equation (5.29) in expectation[4]. The SELU function is given by

[4]To ensure this requirements in expectation the weights will be initialized with values sampled from a normal distribution with a mean of zero and a variance of 1/n where n denotes the dimension of the input vector.

$$\mathrm{SELU}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \ . \end{cases} \tag{5.30}$$

The proposed parameters are α=1.6733 and λ=1.0507. Figure 5.3 illustrates the SELU function with the former constants. To prevent SNNs from overfitting, dropout, as discussed in section 5.3.2, can be applied. In the paper a special method of dropout referred to as alpha dropout is introduced. In this method nodes are randomly set to -$\lambda\alpha$. The scaling of the weights with the dropout parameter p is done at training time. In the paper dropout rates of 0.05 and 0.10 were found to empirically work well. SNNs outperformed various machine learning algorithms on a variety of machine learning tasks with typically very deep architectures.

5.3.4. Hyperparameter Search Methods

Hyperparameter search methods aim to find the best working hyperparameter configuration. Typically the learning rate, weight decay, number of layers, number of nodes per layer, dropout rate etc. are hyperparameters. In contrast weights and bias of a neural network are not hyperparameters as they are derived in the supervised learning cycle which is outlined in section 5.2, rather for every set of hyperparameters the supervised learning cycle has to be carried out. Additionally, the lack of rules to derive the best hyperparameters leads to numerous search algorithms that typically train many hyperparameter configurations and compare their performance. This is done based on the training set splits as outlined in chapter 5.2.2, where the best hyperparameter configuration is the one with the lowest validation set loss. A common approach is to define a discrete hyperparameter space and train every model in order to keep the best performing one. This procedure is denoted as grid search. Another approach is to sample hyperparameters from a distribution, train many models and keep the best model based on the evaluation loss, this is referred to as random search. In contrast to grid search this procedure will search in a bigger hyperparameter space given the same computational budget. Yet another approach is based on Bayesian statistics. The aim of this approach is to refine the distribution of the evaluation loss given the hyperparameters with the Bayesian theorem in conjunction with the performance of former trained configurations. This work uses gird search as well as a state of the art random search algorithm hyperband that is based on successive halving.

Successive Halving

Successive halving samples n sets of hyperparameter configuration based on given distributions. Every configuration will be trained for r epochs and evaluated. The half best performing configurations remain and get trained for another r epochs. This procedure will be repeated for many iterations until only one configuration is left. Following the discussion seen in [49] it is a-priori unclear if successive halving should be carried out with a high number of initial models n and a low number of training steps r or vice versa given the same computational budget. As seen in figure 5.5 discrimination between two

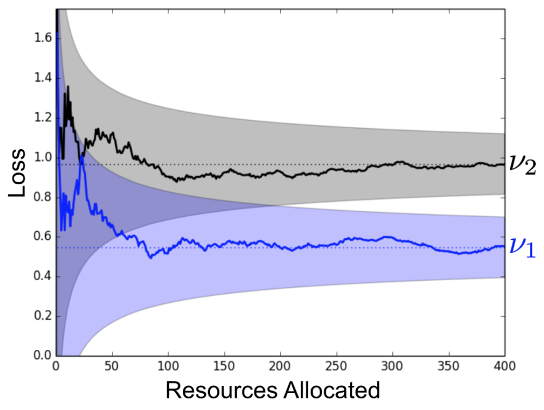


Figure 5.5.: Typical evaluation loss of two hyperparameter configurations over the number of epochs. The shaded areas correspond to the maximum loss deviation per model. This illustration is taken from [49].

hyperparameter configurations can only be achieved at the epoch where the maximum loss deviations do not overlap. Therefore a bigger per model loss deviation as well as similar evaluation loss values would favor a smaller n and a bigger r and vice versa. As the evaluation loss depending on the training epoch is not known in advance hyperband introduces a progression scheme for n and r, hence many successive halving runs with different parameters are carried out as follows.

Hyperband

The hyperband algorithm has two inputs, namely R which is the maximum number of training epochs for one configuration and η which controls the number of thrown out configurations at each iteration of a successive halving run. Based on the former two input parameters the maximum number of successive halving runs is determined by $\mathrm{s}_{\mathrm{max}}+1$ with $s_{max} = \lfloor \log_\eta R \rfloor$. The inputs for the $\mathrm{s}_{\mathrm{max}}+1$ successive halving runs are given by $n = \lceil \frac{s_{max}+1}{s+1} \eta^s \rceil$ and $r = R\eta^{-s}$ as a function of $s \in \{s_{max}, s_{max}-1, \dots, 0\}$, this is the outer loop. The first successive halving run with $\mathrm{s}=\mathrm{s}_{\mathrm{max}}+1$ leads to a maximum number of sampled configuration which are trained for the fewest amount of epochs controlled by n and r respectively. As s decreases less configurations are sampled but the configurations are trained for more epochs. Approximately the overall number of epochs is equal for every successive halving run. Based on n and r in a successive halving run n configurations are sampled that are trained for $r_i = r\eta^i$ epochs keeping the $n_i = \lfloor n\eta^{(-i-1)} \rfloor$ best performing configurations as a function of $i \in \{0, \dots, s\}$, this is the inner loop. Hence a progression of r_{i} and n_{i} for every iteration of a successive halving run is provided in an inner loop, additionally the progression of the inputs of all successive halving runs given by r and n are provided in an outer loop. In the publication hyperband parameters of η=3 and R=91 are proposed to suit well for a variety of tasks and the progressions for the former parameters are summarized in table 5.1. Hyperband

Table 5.1.: Summary of the different successive halving runs that are carried out in a hyperband run with η=3 and R=91 as input values.

	s=4		s=3		s=2		s=1		s=0	
i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

can be applied on many machine learning tasks differing in the definition of resources. The above explanation uses training epochs as resources based on supervised training of neural networks. Lastly hyperband outperformed standard Bayesian hyperparameter search algorithms in various experiments as seen in [49].

5.3.5. Neural Network Architectures

In chapter 5.1 the fully connected feed forward neural network with its fundamental structure seen in equation (5.1) was introduced. This simple architecture lacks many abilities, among others it cannot deal with data of the same structure but in different locations such as faces in different locations of a photograph[5] nor can it deal with input vectors of different dimensions[6]. In the following chapters two different neural network architectures are introduced. All the other concepts such as the optimization of the parameters, regularization, different activation functions and hyperparameter search remain unchanged. Only the simple structure of a FNN given by equation (5.1) and hence the gradient form in the backpropagation are adapted.

Convolutional Neural Network

CNNs (Convolutional Neural Networks) use as well a stacked structure of different layers where the output of one layer is the input of the following layer, hence the information is fed forward. But in contrast to fully connected layers seen in equation (5.9) CNNs use convolution layers instead. A convolution is defined as

$$(f * g)(\tau) = \int_{\mathbb{R}} f(t)g(-(t-\tau))dt \ . \tag{5.31}$$

In equation (5.31) f and g are continuous functions, the first minus sign in the argument of g corresponds to mirroring this function which leads to the property $f * g = g * f$,

[5]In principle a FNN could deal with this kind of data if every possible translation would be present in the training data, this would lead to a very large dataset. However, a CNN (Convolutional Neural Network) is much better suited for this task.

[6]A LSTM (Long-Short Term Memory Neural Network) is designed to deal with that situation.

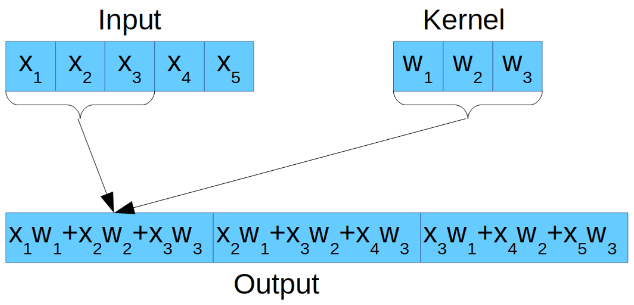

Input
x_1 x_2 x_3 x_4 x_5
Kernel
w_1 w_2 w_3
$x_1w_1+x_2w_2+x_3w_3$ $x_2w_1+x_3w_2+x_4w_3$ $x_3w_1+x_4w_2+x_5w_3$
Output

Figure 5.6.: Illustration of a convolution layer. Five input features are convolved with a kernel given by three weights. The convolution is valid meaning that only values are taken into consideration where the kernel fully overlaps with the inputs resulting in a feature map consisting of three outputs. Usually a bias term is added to every output.

this is denoted as commutativity. As the input of a neural network is discrete and the mirroring of g is only useful for mathematical proofs a CNN uses the following operation

$$(f * g)(n) = \sum_{m=-\infty}^{m=\infty} f(m)g(m+n) \; . \tag{5.32}$$

The definition of equation (5.32) would be a discrete cross-correlation operation if f would be complex conjugated. However, the above formula is the one applied in a CNN[7] and is loosely referred to as convolution even though it is technically a cross-correlation. The discrete function f of equation (5.32) is denoted as the input, g is the discrete kernel or filter and hence the learnable parameters, finally (f ∗ g)(n) is the output which is denoted as the feature map. The former equation represents a one dimensional convolution, it can easily be extended to more dimensions according to the form of the input data[8]. Figure 5.6 illustrates a valid 1D convolution. In this figure five input features are convolved with a kernel represented by three weights leading to a feature map with three outputs. Valid connotes that only values with a fully overlapping kernel are computed. A cross-correlation is typically used to measure the similarity between a given pattern represented by the kernel and the input data spatially. The output is maximized as the kernel overlaps with the fraction of the input data which is nearly of the same shape as the kernel. For example the kernel represents the pixels of a head in a picture. After the convolution high values in the feature map would suggest that indeed a head is present in the picture, even more the spatial position of the face in the picture would be derived. Furthermore, the following three properties of a CNN are from great importance following the discussion seen in [39].

Sparse Interactions The kernel of a convolution layer is smaller as the input. This leads

[7]A convolution layer has typically many feature maps calculated with different kernels. Additionally, every kernel carries a bias term. This bias is a constant added to the form seen in equation (5.32).

[8]For example a 1D convolution is suitable for time series data and a 2D convolution is commonly used for image processing.

to the fact that not all input features are directly connected to every output node which is the case in a FNN. This property leads to smaller models and secondly a convolution layer looks on smaller peaces of the input data and therefore extracts features on smaller scales. This is very well suited to image recognition as details of a picture could give more insights contrary to the whole picture. Lastly if many convolution layers are stacked the lower layer is indirectly connected to a large fraction of the inputs, thus CNNs look at smaller structures per layer and the interaction of those smaller structures over many layers.

Parameter Sharing In a FNN every parameter is used once per forward pass. In contrast a CNN uses the same kernel for the whole input. This leads to a significant decrease in the size of the model.

Equivariant Representation Equivariance is defined as f(g(x))=g(f(x)). This property ensures that if a feature is present in the data in different locations the resulting feature map will be the same but shifted. For example in time series data, if a special signal form appears in the time series at different time stamps the convolution operation detects those special signals but in different locations in the feature map. Therefore smaller shapes in a signal can be detected efficiently. It is noted that a convolution is not capable of detecting other transformations such as rotation, scaling etc..

Many details can be specified in a convolution layer. Padding means that the input signal will be extended with zeros in order to get a specific feature map size. As the convolution is valid the size of the feature map will be smaller as the input size, zero-padding can prevent this. The stride determines the movement of the kernel. A stride of one means that the kernel will convolve over the input jumping only by one input feature. Based on the former thoughts the size of the feature map evaluates as

$$N = \frac{I - K + 2P}{S} + 1 \ . \tag{5.33}$$

In equation (5.33) N is the size of the feature map, I is the size of the input, K is the size of the Kernel, P is the size of zero padding applied at the end and the beginning of the input and S is the stride. As an example a convolution applied to a signal of ten input features with a kernel size of five, a stride of one and no zero padding would consequently result in a feature map size of six.

In summary a convolution layer applies n valid convolutions across the input data with n different kernels as seen in equation (5.32) where additionally per kernel a bias is added resulting in n feature maps with a size calculated by equation (5.33). The output feature maps of one convolution layer can be the input of a following convolution layer or a following fully connected layer. A usual CNN consists of one to three convolution layers leading into one to three fully connected layers as suggested in [42]. As with FNNs every element of the feature map is the argument of an activation function and the standard methods such as dropout, batch norm, selu, etc. are used. The former introduction briefly summarized the concept of CNNs a detailed discussion is given in [39, 42].

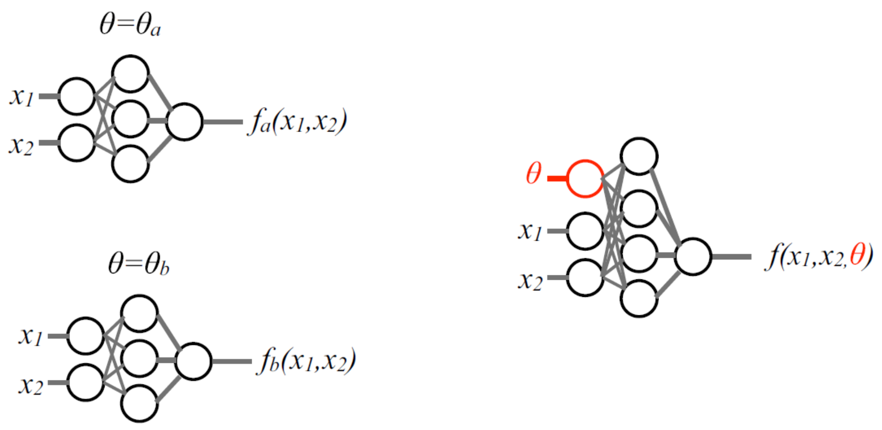


Figure 5.7.: Depiction of two distinct approaches to deal with an overlying parameter present in the data. One approach is to train a neural network for every value of that parameter, this is depicted on the left side. On the right side one FNN with an additional input for the parameter is trained, this is referred to as a PNN. This illustration is taken from [50].

Parameterized Neural Network

A PNN (Parameterized Neural Network) is designed to deal with data following a similar structure but differing based on a single parameter. In a physical context it may be the same physical process with different parameters. For example elastic scattering processes with different total energies. Another use case of PNNs is if the distribution of training data differs from the distribution of the real data depending on that parameter. Again considering elastic scattering with training data consisting of five sample with a given energy and three samples with another given energy. This would be a kind of prior determining the distribution of the training data based on the overlying parameter. A machine learning algorithm is likely to learn that the data is always provided with the same distribution as in the training data, hence it is biased. Many ways of dealing with this kind of parameter are possible. One concept follows the idea of training a neural network for every value of the parameter and averaging the output of all trained models at evaluation time. Depending on the parameter a large amount of neural networks have to be trained. A different concept is to extend the input vector of the neural network with a single entry representing the overlying parameter itself, this is referred to as a PNN. The former two concepts are illustrated in figure 5.7. PNNs are investigated in [50] and summarized above. Considering CNNs, extending the inputs with the respective parameter may result in unwanted effects. However, the additional parameter could be introduced at the stage of fully connected layers.

6. Binary Classifier Evaluation

A binary classifier labels every input with one number representing the predicted class. It is from great interest how the algorithm performs, hence many ways of evaluating a binary classifier are available. A first impression can be given by the value of the loss function for the evaluation data. Another approach is based on the outputs that can be provided in two ways depending on the algorithm. On the one hand it can be a number ranging from zero to one interpreted as a probability p of the respective sample corresponding to class one and hence (1-p) is the probability for that sample belonging to the class two. On the other hand the output can only be a binary number that assigns every sample a class label without the framework of probabilities. In the latter case a usual measure is the score which is defined as the percentage of the data that is labeled right relative to all evaluated samples. The score can also be derived for classifiers in the probability framework if every output above a certain limit is interpreted as class one and vice versa for the other class. This cut limit is a real number between zero and one. After a cut is applied the following quantities can be defined.

Positives (P) are all samples in the dataset that belong to the class which is defined as positive.

Negatives (N) are all samples in the dataset that belong to the class which is defined as negative.

True Positives (TP) is the number of samples which are labeled as positive and are indeed positive.

False Positives (FP) is the number of samples which are labeled as positive but are negative. Hence labeled wrong. In a statistical framework a FP is a Type I error meaning that the true null hypothesis (negative sample is indeed negative) is rejected.

True Negatives (TN) is the number of samples which are labeled as negative and indeed are negative.

False Negatives (FN) is the number of samples which are labeled as negative but are positive. In a statistical framework this is denoted as Type II error meaning that a false null hypothesis (negative sample is positive) is accepted.

True Positive Rate, Recall (TPR) is defined as the percentage of right labeled positives TP relative to all positives P present in the dataset. With the formal definition of $TPR = TP/P = TP/(TP + FN)$. In a physical context the positive samples are signals that represent the wanted physics. Thus the TPR is referred to as the signal surviving probability or signal efficiency. Another denotation for the TPR is recall which answers the question how many positives are labeled as positive.

False Positive Rate (FPR) is the percentage of samples that are wrong labeled negatives FP relative to all negative samples N in the dataset. The definition is $FPR = FP/N = FP/(FP + TN)$. In a physical context negatives represent background that is not from interest for the wanted physics. Hence the FPR is called background surviving probability or background leakage.

Precision is defined as the percentage of right labeled positives TP relative to all samples labeled as positive with the formal definition of $Precision = TP/(TP + FP)$. Precision answers how many of the labeled positives are indeed positive.

The above quantities are a small selection of existing values that are used to evaluate a binary classifier, these are chosen due to their use in this present work. A more comprehensive list can be found at [51]. As the above values are depending on the cut limit it is from great interest to illustrate this dependency graphically. This is usually done in two ways as follows.

6.1. Confusion Matrix

A confusion matrix plots the TP, FP, TN and FN in matrix form with a color scale to visualize the performance of the classifier. Figure 6.1 shows the illustration of a confusion matrix. Due to the color scale it serves as a fast visualization of the classifier. However, for every cut limit a confusion matrix has to be created.

6.2. ROC Curve

The ROC Curve (Receiver Operating Characteristic Curve) plots the TPR versus the FPR or in physical terms the signal efficiency over the background efficiency as a function of the cut limit, hence the ROC curve illustrates the classifier performance for every cut limit. The best possible classifier reaches the point (0,1), thus every sample is labeled right. A random discriminator would be represented by a line with a slope of one and no offset. A cut limit of zero corresponds to the point (1,1). This point is reached because every sample would be labeled as positive thus the TPR as well as the FPR equals one. The point of (0,0) is reached with a cut limit of one with an analogous explanation. Figure 6.2 shows a sample ROC curve. A detailed discussion of the information present in ROC curves can be found in [52].

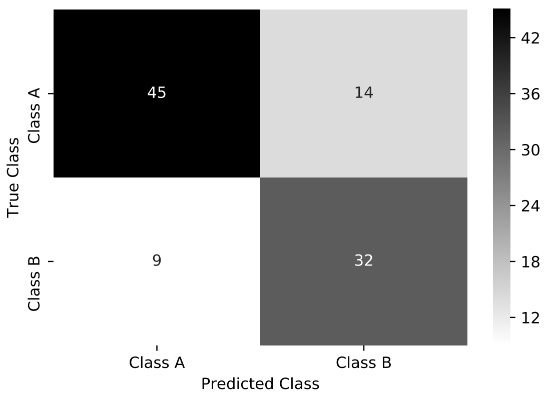


Figure 6.1.: Illustration of a confusion matrix for a binary classification problem. It illustrates the true positives TP, false positives FP, true negatives TN as well as the false negatives FN in matrix form for a given cut limit. Based on the confusion matrix further quantities such as precision, recall, false positive rate etc. can be evaluated. The color scale provides a strong visualization of the performance of the model.

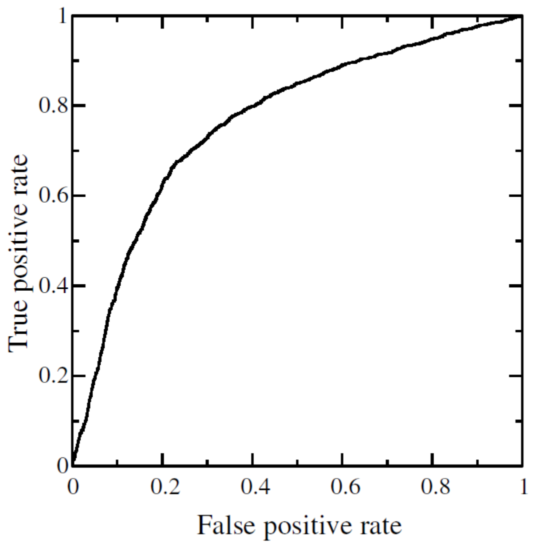


Figure 6.2.: Illustration of a ROC curve. The true positive rate TPR and the false positive rate FPR are evaluated based on the output of a classifier for different cut limits ranging from zero to one. A perfect classifier would reach the left upper corner. This illustration is taken from [52].

7. Results

In chapter 3 the TUM40 detector with its two different pulse shapes and the resulting classification problem was introduced. As this is a supervised learning task labeled training data are required. The favored method of generating artificial pulses as well as the three distinct generated datasets are outlined in section 4, a brief summary and the nomenclature used in this chapter is given in table 7.1. These datasets are used to train different neural network architectures in order to solve the TUM40 classifications task. The training procedure is exactly done as described in chapter 5.2 with the following specifications. In order to obtain a class probability as an output, every neural network has one output node with a Sigmoid activation function, moreover the cross-entropy loss function given by equation (5.4) is used. The targets are defined as follows, absorber pulses are labeled with one and carrier pulses are labeled with zero. Every dataset used to train the models is randomly divided into three distinct sets. 10000 pulses are used to train the network, 5000 pulses are used to evaluate the network at every training epoch and the remaining pulses are used to test the networks as training is finished, hence every outlined result below is referring to the unseen test data of the respective dataset. A single pulse consists of 8192 samples which determines the number of inputs. Additionally, PNNs are fed with the pulse as well as the pulse height parameter discussed in section 2.3.1 resulting in 8193 input features. In chapter 5.2.2 methods of standardizing every input feature are discussed, however the raw pulses are fed into the neural network and the argumentation for this decision is given in appendix A. Training is done in Python with the neural network framework PyTorch [53, 54] that provides the backpropagation algorithm as well as many different neural network architectures and everything else related to deep learning. Many experiments were made with the ADAM and the NAG optimization algorithm described in section 5.2.4, the ADAM algorithm showed the overall best performance, thus this algorithm is exclusively used. For regularization described in section 5.3.2 dropout and L2 were used and modeled as hyperparameters, additionally, dataset splits in conjunction with the early stopping algorithm were implemented. In order to derive meaningfully hyperparameters the hyperband algorithm and the grid search algorithm were used as outlined in section 5.3.4, the best model was determined by the lowest evaluation loss, furthermore the corresponding hyperparameter spaces are defined in the following respective chapters. Intermediate results for FNNs (Section 5.1), PNNs and CNNs (Section 5.3.5) are presented. Based on the intermediate results a final neural network model is derived and used to analyze the real measured CRESST data. A short summary of the real measured data is given in appendix B. Lastly the methods discussed in section 6 are used to evaluate the performance of the neural networks.

Table 7.1.: Nomenclature and properties of the three generated datasets described in section 4.3 which are used to train and evaluate different neural networks.

	Dataset 1	**Dataset 2**	**Dataset 3**
Template	x	-	x
Analytic Fit Function	-	x	-
Time Shift	-	x	-
No Time Shift	x	-	x
Measured Noise	x	x	-
Simulated Noise	-	-	x
Section	4.3.1	4.3.1	4.3.2

Table 7.2.: Summary of the hyperparameter space used to find the best performing model for **FNNs** as well as **PNNs**.

Hyperparameter	Description
n_{Layer}	Number of fully connected layers
n_{Nodes}	Number of the nodes per hidden layer
progression	This binary parameter determines if the number of nodes per hidden layer is constant or exponentially decays with a basis of two
p	Dropout rate
n_{Batch}	Size of the mini batch
wd	Parameter that determines the strength of the L2 weight decay

7.1. Results of FNNs

In a first step FNNs were trained and evaluated on dataset 1. Based on the hyperparameter space seen in table 7.2 the model depicted in figure 7.1 showed the best overall evaluation loss. Figure 7.2 (a) displays the outputs of the neural network versus the pulse height parameter of every pulse. The upper histogram represents all generated absorber events and the lower histogram represents all generated carrier events. Figure 7.2 (b) shows the corresponding ROC curves for energies that are above and below 0.8 keV[1] based on the pulse height parameter. In figure 7.2 (c) the confusion matrix for a cut limit of 0.5 is seen. From about 200000 pulses only about 350 are falsely labeled at the former cut limit. This is a great performance. In the histograms of the outputs,

[1]In section 2.3.3 the energy reconstruction based on the amplitude of the standard event fit is discussed. The pulse height parameter is an estimator of the standard event fit amplitude, therefore it can be used to reconstruct the energy of absorber events (not valid for carrier events as an energy calibration is only done for absorber events, see also section 4.1). However, at low energies the pulse height is likely to derive an amplitude completely dominated by noise. Hence all reconstructed energies based on the pulse height parameter act as a rough guideline. In section 4.3 the connection between the pulse height parameter and the scale factor is outlined.

perfect discrimination is observed in higher energy regions, in lower energy regions falsely labeled pulses are seen. This is due to the fact that at lower energies the amplitude of the pulses is of the order of the noise which makes the classification more challenging.

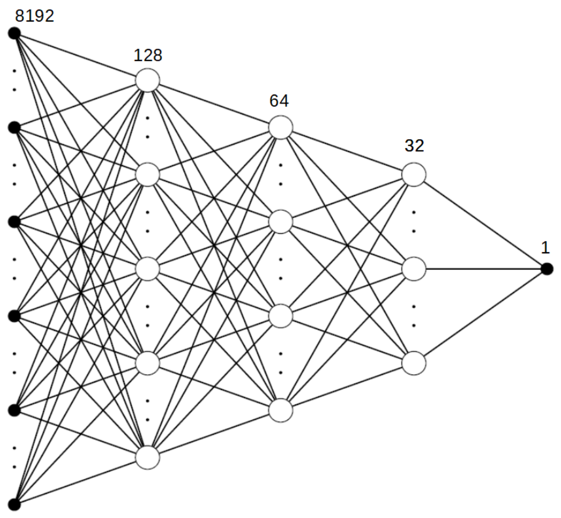
Figure 7.3 displays the performance of the neural network architecture seen in figure 7.1 trained on dataset 1 evaluated on dataset 2. As observed the performance significantly worsened compared to figure 7.2 as the evaluation dataset contains time shifted events and the training set does not. Feed forward neural networks are in principle not capable of working with shifted data as discussed in chapter 5.3.5, thus a convolutional neural network would be more suitable. However, the performance could be improved by training on dataset 2 as outlined below.

In figure 7.4 the performance of the neural network architecture seen in figure 7.1 trained on dataset 1 evaluated on dataset 3 is depicted. This dataset contains pulses of much higher energies as the dataset which was used to train the neural network. As a result the discrimination works well in the covered energy region of the training set but at higher energies every pulse is labeled as an absorber event. This behaviour was observed in many trained models and shows that neural networks are usually not capable of dealing with data of similar shape but with differences in the scaling. One way of dealing with this problem would be to include all pulse of all scales which are expected to be in the real data. However, at higher energies the pulse shape anyhow differs as the TES saturates.

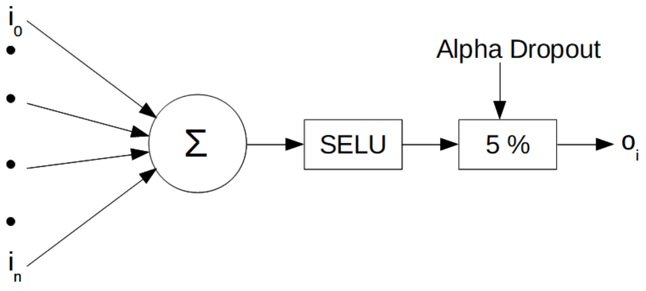
In another experiment the neural network architecture seen in figure 7.1 was trained and evaluated on dataset 2. The performance of the model is seen in figure 7.5. Comparing the confusion matrix of figure 7.5 with 7.2 an one order of magnitude increase of falsely labeled pulses is observed. This rather small increase indicates that a shift in the pulses of 20 ms as present in the training data (dataset 2) is a rather insignificant translation, proven by the fact that a FNN can learn data with translation if enough variety is present in the training data. Typically that results in very large datasets but in the former case 10000 pulses seem to be enough. Nevertheless CNNs will be examined in the later chapters.

7.2. Results of PNNs

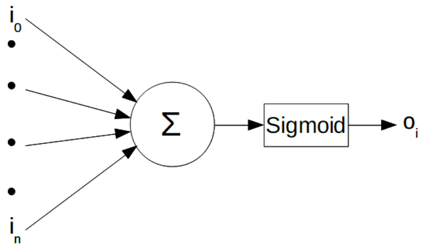
In the former chapter the performance of a FNN was discussed in detail. It is observed that a FNN performs very well on non time translated data and even performs acceptable on time-shifted pulses. However, an overlying parameter of the pulses is the amplitude determined by the deposited energy. In the pulse generation procedure amplitudes were uniformly sampled, hence the arbitrary assumption that the energies of the real data are uniformly distributed is implicitly modeled in the training data. Consequently a parameterized neural network is expected to enhance the performance as discussed in section 5.3.5. Two possibilities are available to model the overlying parameter. Firstly the pulse could be fit with the template and the resulting amplitude would act as the parameter. This approach possesses many drawbacks. It is a-priori not clear which template should be taken, additionally, the template fit is an extensive procedure as described in [30] and moreover this work aims to skip the fitting procedure in the pulse



(a) Visualization of the whole **FNN** architecture.

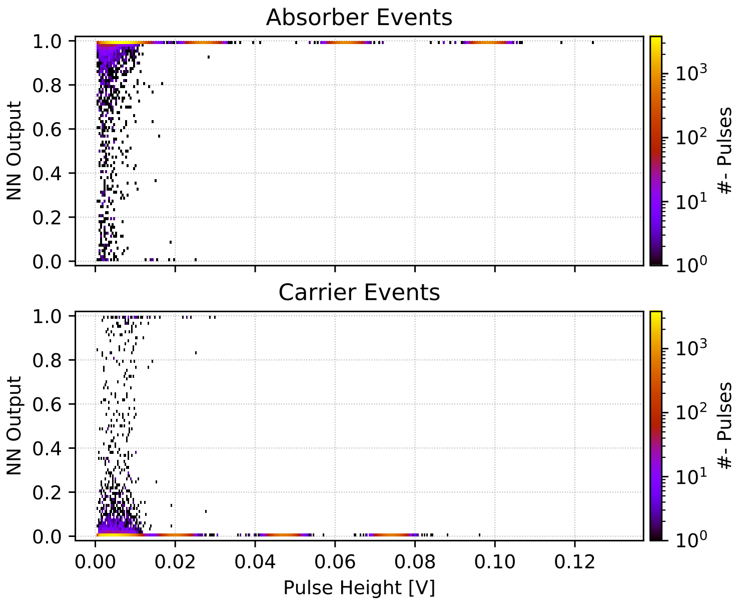


(b) Structure of all hidden nodes.

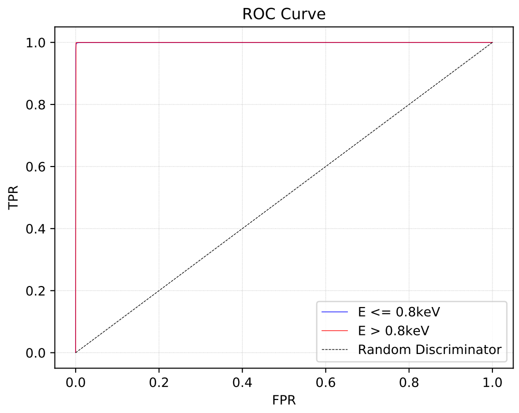


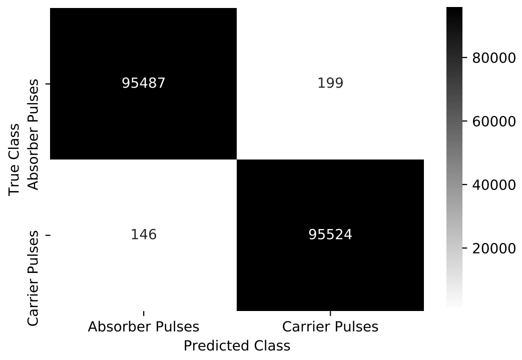
(c) Structure of the output node.

Figure 7.1.: Model of the **FNN** architecture trained on different datasets. (a) Overview of the model with all layers and the number of nodes per layer. (b) Structure of every hidden node. (c) Structure of the output node. This model is trained with the ADAM optimizer with a mini batch size of 50, a binary cross-entropy loss function and a L2 weight decay parameter of 0.0001.



(a) Histograms of the neural network outputs as a function of the pulse height parameter.

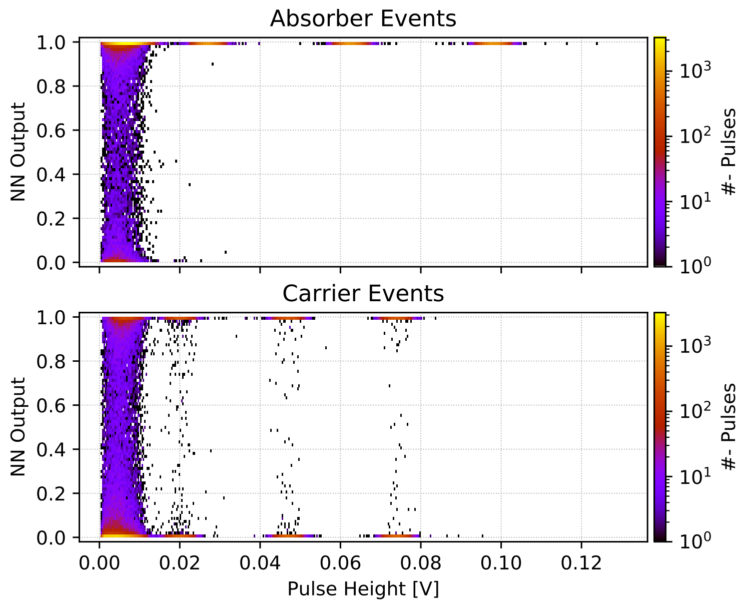




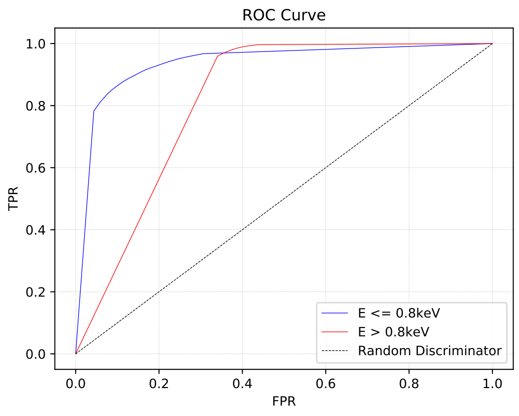
(b) ROC curve for the test data.

(c) Confusion matrix for the test data at a cut limit of 0.5 on the neural network output.

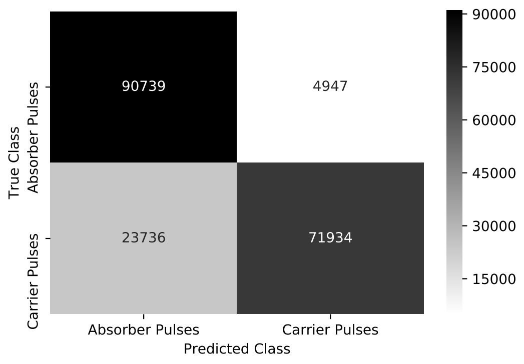
Figure 7.2.: Results of the **FNN** presented in figure 7.1 which was **trained and evaluated on dataset 1**. (a) Output of the neural network is plotted against the pulse height parameter. The upper histogram displays all generated absorber events and the lower histogram displays the outputs for all generated carrier events. (b) ROC curve of the test data for energies above and below 0.8 keV based on the pulse height parameter. (c) Confusion matrix for the test data. Values are calculated for a cut limit of 0.5 on the neural network output.



(a) Histograms of the neural network outputs as a function of the pulse height parameter.



(b) ROC curve for the test data.



(c) Confusion matrix for the test data at a cut limit of 0.5 on the neural network output.

Figure 7.3.: Results of the **FNN** presented in figure 7.1 which was **trained on dataset 1 and evaluated on dataset 2**. (a) Output of the neural network is plotted against the pulse height parameter. (b) ROC curve of the test data for energies above and below 0.8 keV based on the pulse height parameter. (c) Confusion matrix for the test data. Values are calculated for a cut limit of 0.5 on the neural network output.

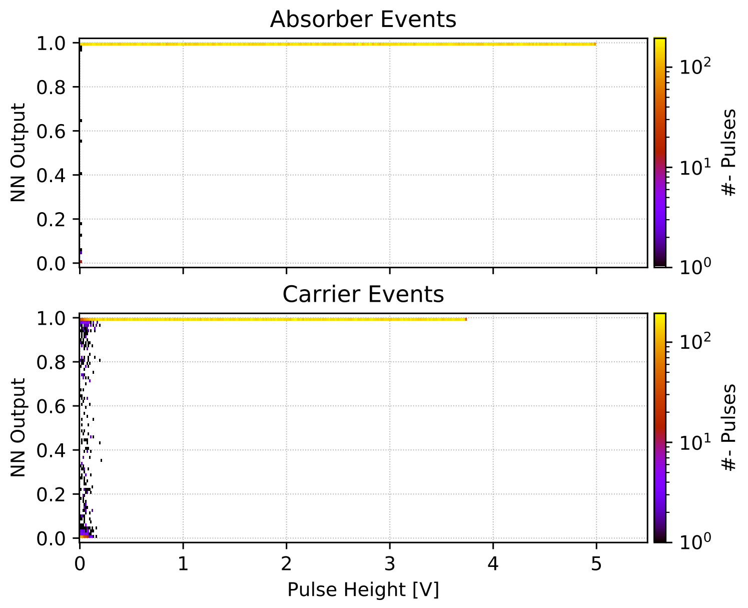


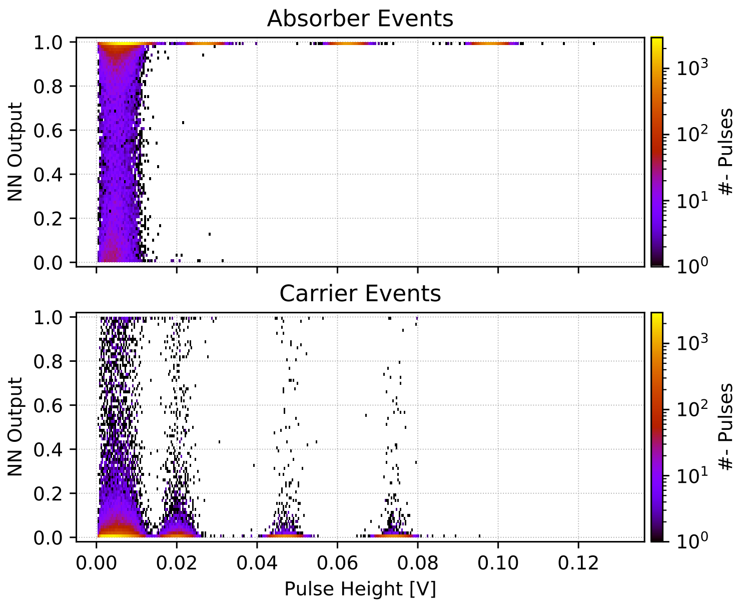
Figure 7.4.: Output histograms of the **FNN** seen in figure 7.1 which was **trained on dataset 1 and evaluated on dataset 3**. The output of the neural network is plotted against the pulse height parameter for absorber and carrier events.

shape discrimination contrary to the methods used before as summarized in section 3.2. The second approach is to evaluate the pulse height parameter as seen in section 2.3.1 and use it as an estimator of the amplitude. This can be done very easy for training as well as real data and does not require a template. However, as discussed in section 4.3.1 the pulse height underestimates the amplitude of carrier pulses and slightly overestimates the amplitude of absorber pulses. Another issue is that at low energies both approaches can measure an arbitrary amplitude originating from the noise. This leads to a pulse height which only gives a rough idea about the magnitude of the amplitude at low energies. Due to simplicity the pulse height parameter is chosen adding an additional dimension to the input data fed into the neural networks resulting in 8193 input features.

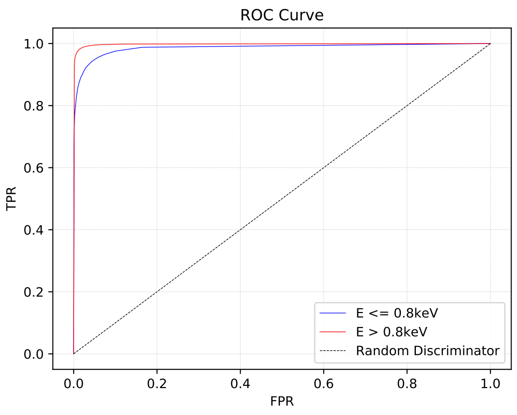
Figure 7.6 depicts the best performing models based on the hyperparameter space seen in table 7.2. This model was trained and evaluated on dataset 1 and the results are illustrated in figure 7.7. The performance of the PNN is comparable to the performance of the FNN seen above with about only 350 pulses falsely labeled. However, because of the issues with the pulse height parameter discussed above and moreover the implementation in a CNN architecture is ambiguous this approach is skipped for further models.

7.3. Results of CNNs

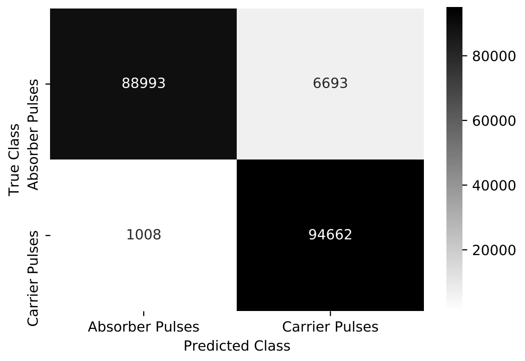
In the former chapters FNNs as well as PNNs were trained and the performance was examined with the conclusion that due to the pulses shifted in time a CNN is needed. In



(a) Histograms of the neural network outputs as a function of the pulse height parameter.

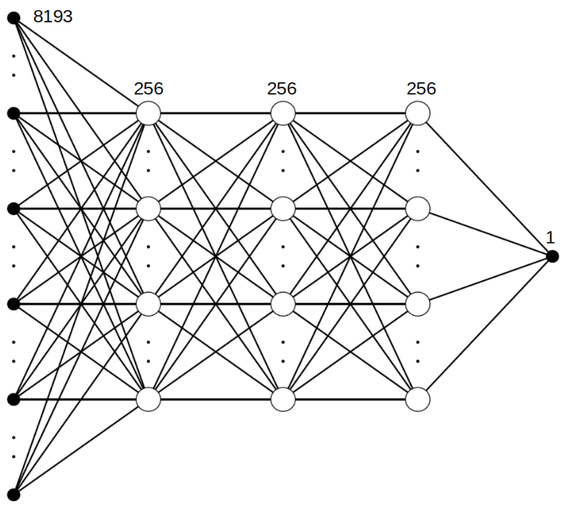


(b) ROC curve for the test data.

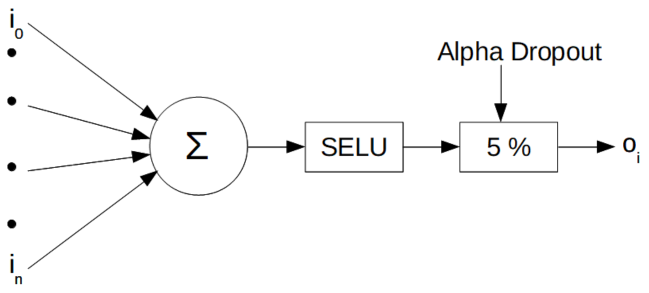


(c) Confusion matrix for the test data at a cut limit of 0.5 on the neural network output.

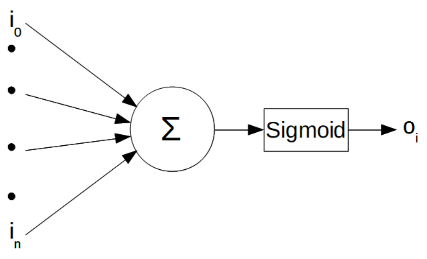
Figure 7.5.: Results of the **FNN** seen in figure 7.1 which was **trained and evaluated on dataset 2**. (a) Output of the neural network is plotted against the pulse height parameter. (b) ROC curve of the test data for energies above and below 0.8 keV based on the pulse height parameter. (c) Confusion matrix for the test data. Values are calculated for a cut limit of 0.5 on the neural network output.



(a) Visualization of the whole **PNN** architecture.

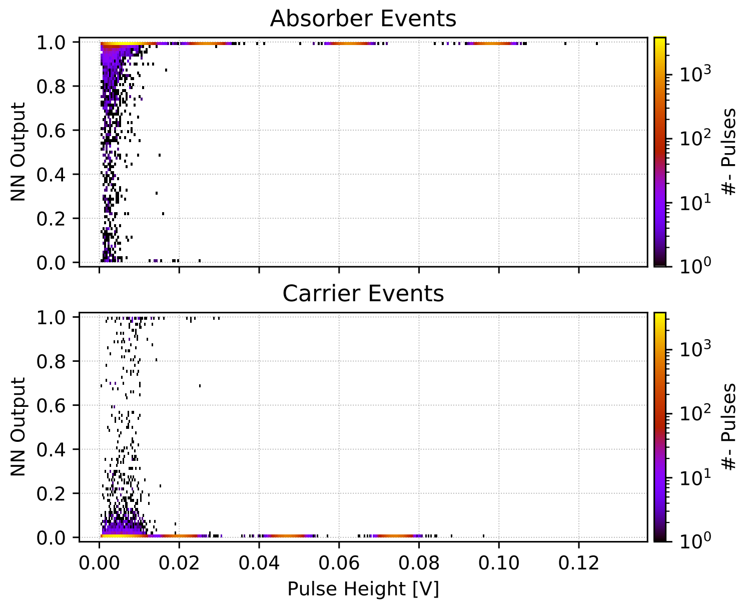


(b) Structure of all hidden nodes.

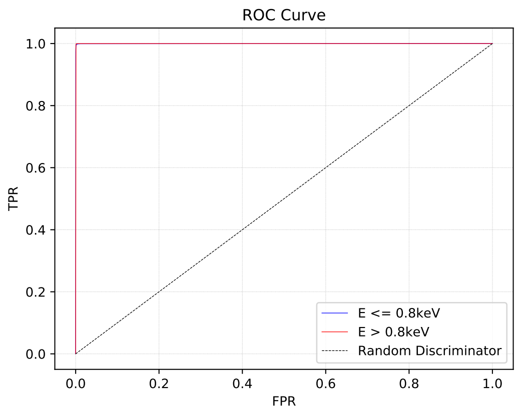


(c) Structure of the output node.

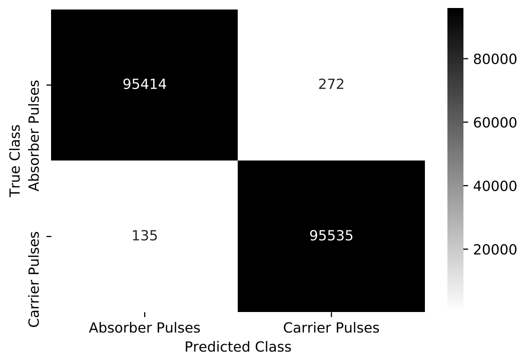
Figure 7.6.: Model of the **PNN** architecture trained on different datasets. (a) Overview of the model with all layers and the number of nodes per layer. (b) Structure of every hidden node. (c) Structure of the output node. This model is trained with the ADAM optimizer with a mini batch size of 50, a binary cross-entropy loss function and a L2 weight decay parameter of 0.0001.



(a) Histograms of the neural network outputs as a function of the pulse height parameter.



(b) ROC curve for the test data.



(c) Confusion matrix for the test data at a cut limit of 0.5 on the neural network output.

Figure 7.7.: Results of the **PNN** seen in figure 7.6 which was **trained and evaluated on dataset 1**. (a) Output of the neural network is plotted against the pulse height parameter. (b) ROC curve of the test data for energies above and below 0.8 keV based on the pulse height parameter. (c) Confusion matrix for the test data. Values are calculated for a cut limit of 0.5 on the neural network output.

Table 7.3.: Summary of the hyperparameter space used to find the best performing **CNN** architecture.

Hyperparameter	Description
n_{LayerCL}	Number of convolution layers
n_{Filters}	Number of filters of the first convolution layer
n_{Prog}	Number that is added to the number of filters of the former convolution layer in order to derive the number of filters for the present convolution layer.
fraction	The convolution layer filter size is derived as a fraction of the input signal length
n_{LayerFC}	Number of fully connected layers
n_{Nodes}	Number of the nodes per fully connected hidden layer
progression	This binary parameter determines if the number of nodes per fully connected hidden layer is constant or exponentially decays with a basis of two
p	Dropout rate
n_{Batch}	Size of the mini batch
wd	Parameter that determines the strength of the L2 weight decay

principle the above lying parameter namely the pulse height as discussed in the former chapter could be introduced in the model as an additional input at the stage of the first fully connected layer. But due to the similar performance of a PNN compared to a FNN as well as the issues regarding the pulse height parameter discussed before, this approach is rejected. In figure 7.8 the best performing model based on the hyperparameter space seen in table 7.3 is illustrated. Dataset 2 is used for training and evaluation. Moreover to boost the performance five different CNNs with the same hyperparamter configuration were trained and their respective outputs were averaged. This approach is denoted as bagging (Bootstrap Aggregating) and averages out the variance present in the outputs if the different used models are uncorrelated. In this case the CNNs are not completely uncorrelated but the diversity originates from the random weight initialization at the beginning of the training.

Figure 7.9 depicts the averaged outputs of the five CNNs trained and evaluated on dataset 2. The performance of the CNN on the time-shifted data is comparable with the performance of a FNN on the non time-shifted data. As seen in the confusion matrix at a cut limit of 0.5 (figure 7.9 (c)) the wrongly labeled pulses are roughly doubled compared to the FNNs performance seen in figure 7.2 (c), still this is a great performance based on the total number of pulses. As expected in the low energy regions the discrimination is problematic due to the signal-to-noise ratio.

To investigate the CNNs further the two filters of the first convolution layer of one model are plotted in figure 7.10. One filter with many oscillations as well as one filter with only one peak are seen. This could indicate that the oscillating filter observes the fast rising carrier like pulse features and vice versa. However, the filters are noisy which could indicate that the weight decay is to low or the number of trained epochs was too low in analogy to the discussion of CNN filters seen in [42].

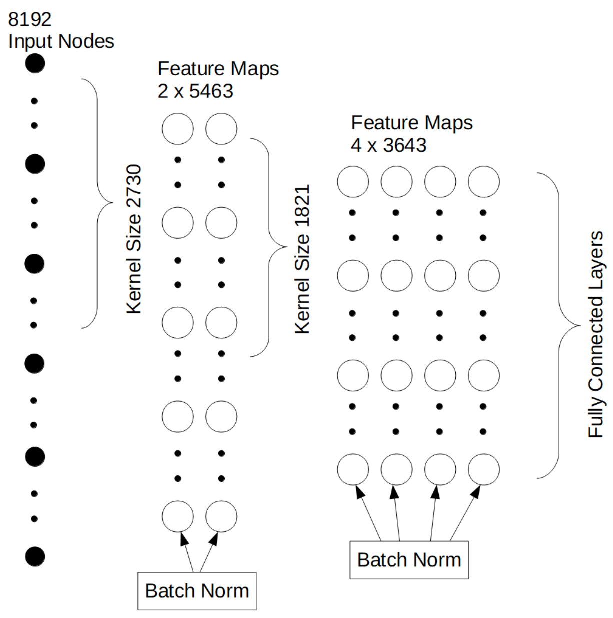
Figure 7.11 shows the outputs of the former described CNN model on the real measured data plotted as a function of the pulse energy given by the standard event fit in combination with the energy reconstruction procedure seen in section 2.3.3. A brief discussion about the real data is given in appendix B. An overview of the whole energy range is seen in figure 7.11 (a). Interestingly at the high energy region six populations at an output of 0, 0.2, 0.4, 0.6, 0.8 and 1 are seen. This can be interpreted as a voting system based on the bagging of five different CNNs. At 0.8 four out of the five CNNs evaluate an output of one whereas one CNN estimates an output of zero leading to an average of approximately 0.8. The training dataset includes only pulses with a maximum amplitude corresponding to 11 keV derived with the pulse height parameter. Thus a similar effect as seen with FNNs in figure 7.4 may be observed in some of the five models leading to that voting effect. However, all five CNNs showed great performance on the dataset with simulated noise (dataset 3). In the low energy region illustrated in figure 7.11 (b) a clear population of pulses labeled with zero are observed. To further investigate the output of the CNNs two cuts are performed which are described in the following chapter.

7.3.1. Cuts

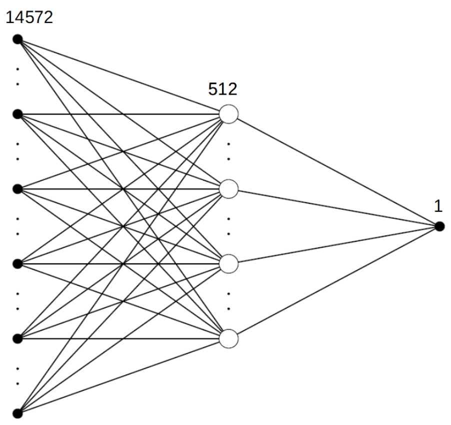
Based on the CNN outputs seen in figures 7.9 and 7.11 two different cuts are applied. Firstly the cut given by the constraint that the number of false negatives equals the number of false positives based on the outputs on the training data (dataset 2) seen in figure 7.9 (**Equal Cut**) is defined. The second cut aims to observe the upper population seen in figure 7.11 (a) which should deliver only absorber pulses with great confidence (**High Cut**). Table 7.4 summarizes the outcomes of the two former introduced cuts and in figure 7.12 both confusion matrices are depicted based on the training data (dataset 2). Figures 7.13 and 7.15 illustrate the energy distributions for both cuts and figure 7.14 and 7.16 represent the light yield plots for both cuts. In the context of the former mentioned figures absorber pulses are all pulses where the CNN output is higher than the cut limit and vice versa for carrier pulses. Both cuts observe the vast amount of carrier pulses at the detector threshold as seen in the light yield plots. This outcome was also observed in [26] and is now confirmed by neural networks directly acting on the raw pulses.

Table 7.4.: Summary of the two cuts based on the neural network outputs seen in figures 7.9 (a) and 7.11 (a). The respective discussion is given in section 7.3.1.

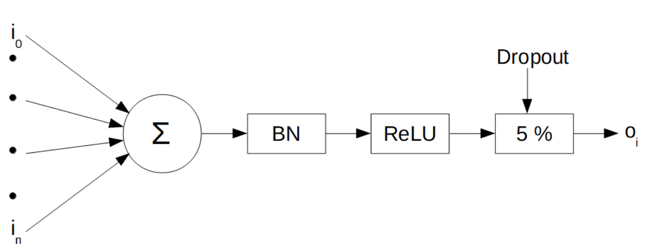
Cut Type	Cut Limit	Total Pulses	Surviving Pulses
Equal Cut	0.3994	89985	70099/77.9%
High Cut	0.98	89985	62601/69.6%



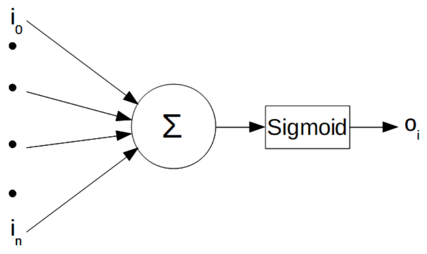
(a) Structure of the convolution layers with a ReLU activation function applied to every node after the batch normalization of the feature map.



(b) Structure of the fully connected layers.

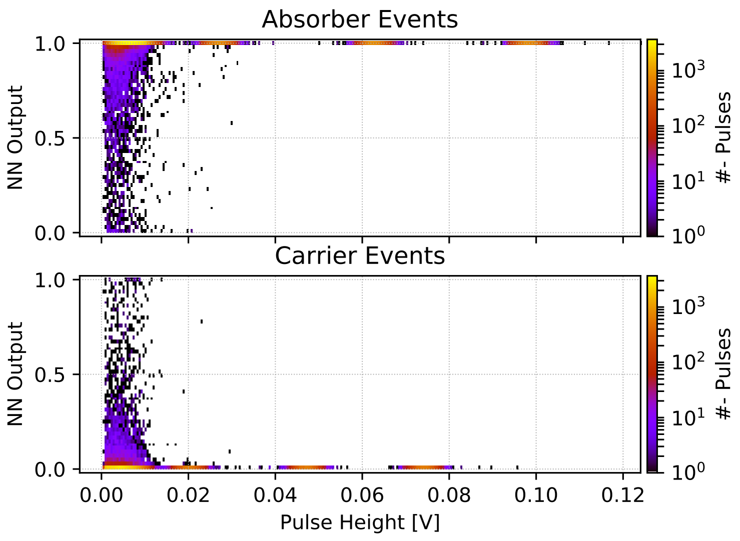


(c) Structure of the hidden nodes of the fully connected layers.

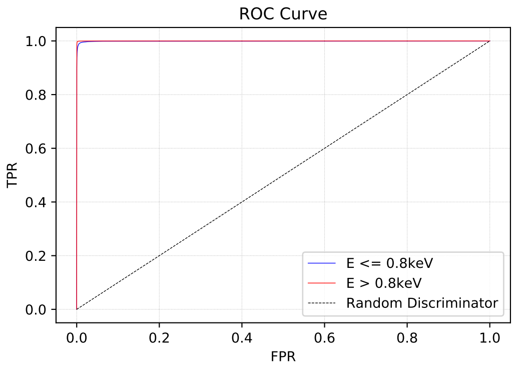


(d) Structure of the output node.

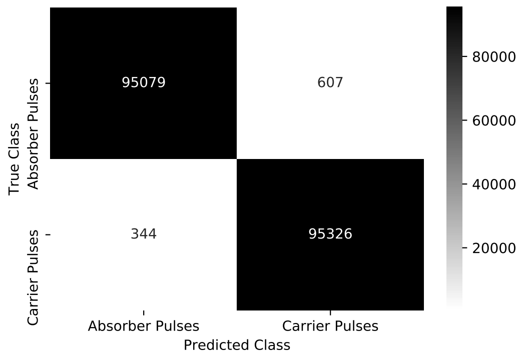
Figure 7.8.: Model of the **CNN** trained on different datasets. (a) Structure of the two convolution layers. No zero padding is applied and the stride is set to one. See also equation (5.33). (b) Structure of the fully connected layers. Data flows from the convolution layers into the fully connected layers. (c) Structure of all hidden nodes of the fully connected layers. (d) Structure of the output node. This model is trained with the ADAM optimizer with a mini batch size of 50, a binary cross-entropy loss function and a L2 weight decay parameter of 0.0001.



(a) Histograms of the averaged neural network outputs as a function of the pulse height parameter.



(b) ROC curve for the test data.



(c) Confusion matrix for the test data at a cut limit of 0.5 on the neural network output.

Figure 7.9.: Results of the averaged outputs of five **CNNs** seen in figure 7.8 which were **trained and evaluated on dataset 2**. (a) Averaged outputs of the neural networks are plotted against the pulse height parameter. (b) ROC curve of the test data for energies above and below 0.8 keV evaluated with the pulse height parameter. (c) Confusion matrix for the test data. Values are calculated for a cut limit of 0.5 on the neural network output.

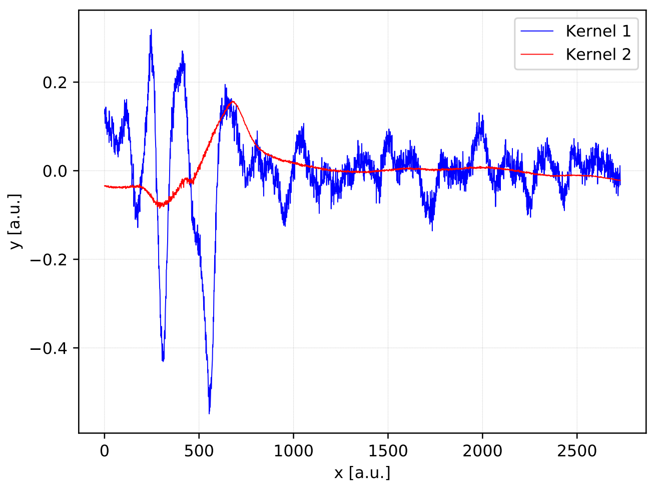
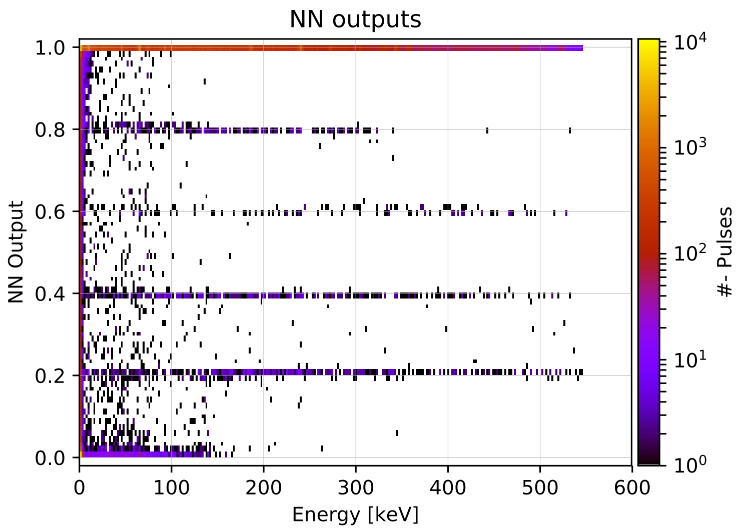
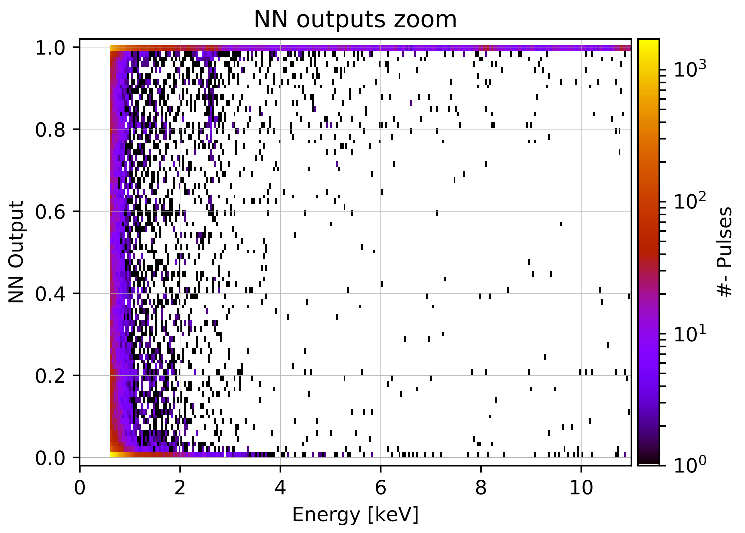


Figure 7.10.: Illustration of the two filters of the first convolution layer from a **CNN** network architecture seen in figure 7.8. This **CNN** was **trained on dataset 2**.

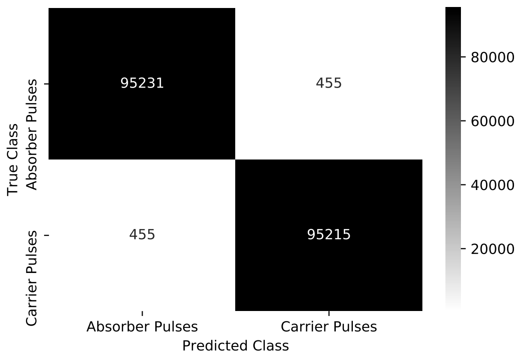


(a) Overview of the averaged neural network outputs on real measured data.

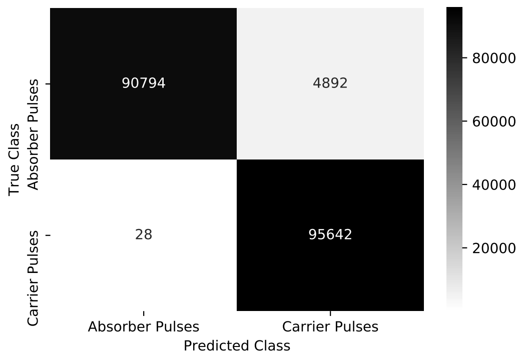


(b) Zoom view of the averaged neural network outputs on real measured data. Zoomed in the energy region which is covered by the training data.

Figure 7.11.: Averaged outputs of five **CNNs** with an architecture depicted in figure 7.8 **evaluated on the real measured data**. The **CNNs** were **trained on dataset 2**. (a) The averaged outputs of the neural networks are plotted as a function of the energy which was determined with the standard event fit (details in [30]). (b) Zoom view into the energy region until 11 keV which is approximately the maximum energy covered by the training data.

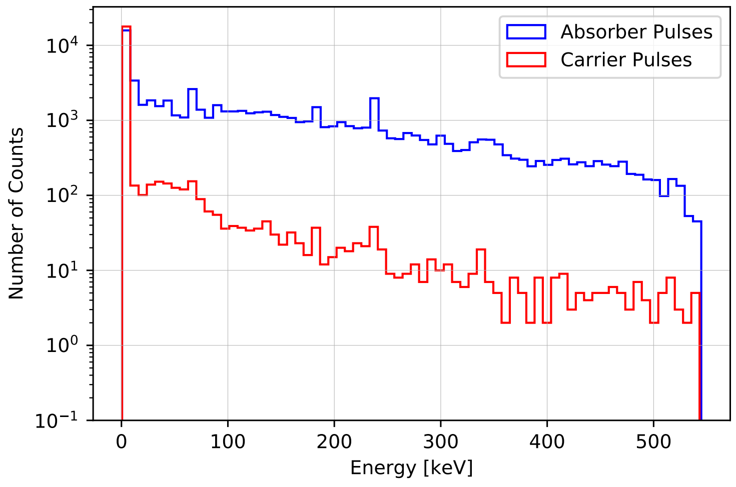


(a) Confusion matrix for the **Equal Cut** with a limit of 0.3994 on the averaged neural network output.

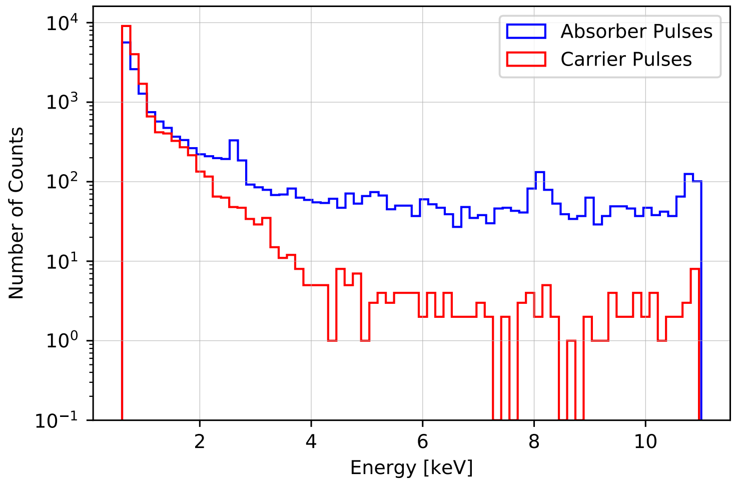


(b) Confusion matrix for the **High Cut** with a limit of 0.98 on the averaged neural network output.

Figure 7.12.: Confusion matrix for the **Equal Cut** and the **High Cut** as summarized in table 7.4. The values are based on the averaged outputs **evaluated on dataset 2** of five **CNNs** seen in figure 7.8 which are **trained on dataset 2**.

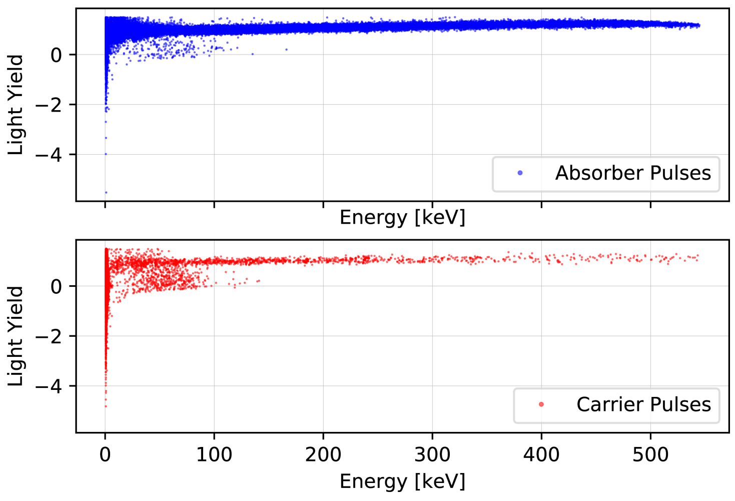


(a) Energy histogram.

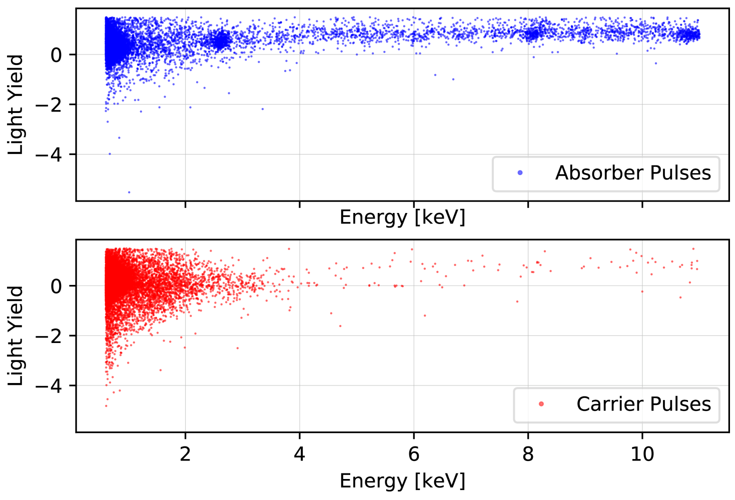


(b) Zoom view of the energy histogram.

Figure 7.13.: Energy histograms after the **Equal Cut** which is summarized in table 7.4. Absorber pulses are all pulses with a neural network output higher than the cut limit. Vice versa for carrier pulses. (a) Energy histogram of all pulses. (b) Zoom view into the region up to 11 keV which is approximately the maximum energy covered by the training data.

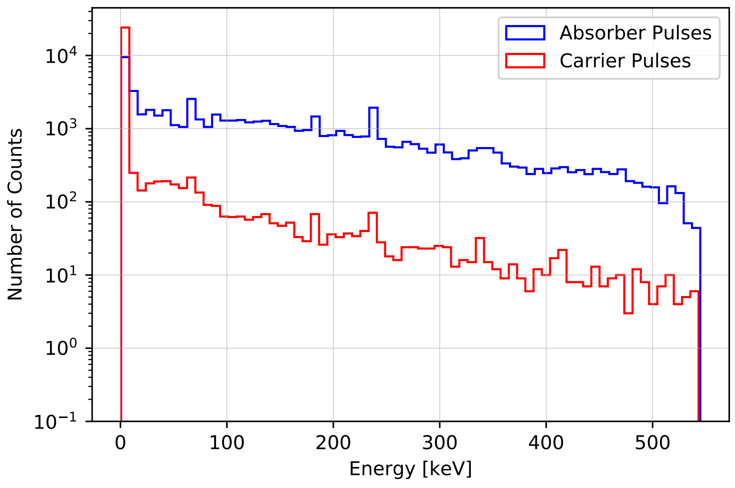


(a) Light yield plot.

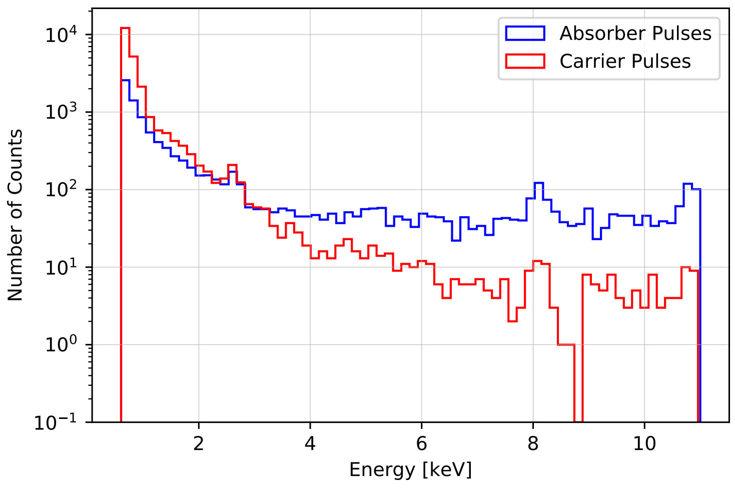


(b) Zoom view of the light yield plot.

Figure 7.14.: Light yield plot after the **Equal Cut** which is summarized in table 7.4. Absorber pulses are all pulses with a neural network output higher than the cut limit. Vice versa for carrier pulses. (a) Light yield plot of all pulses. (b) Zoom view into the region up to 11 keV which is approximately the maximum energy covered by the training data.

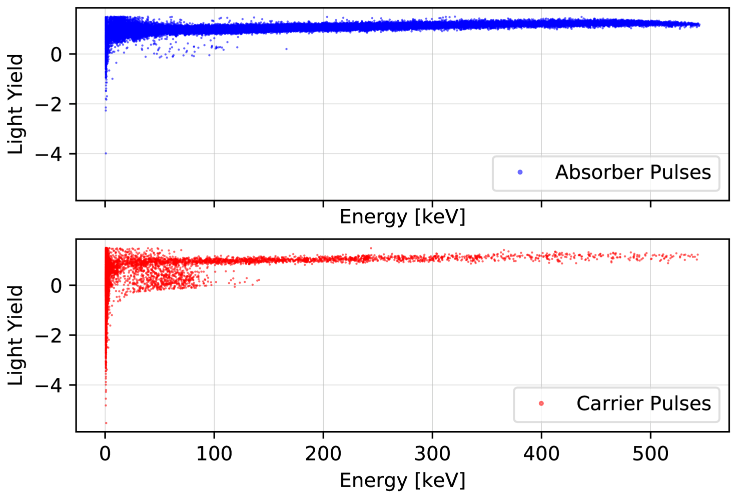


(a) Energy histogram.

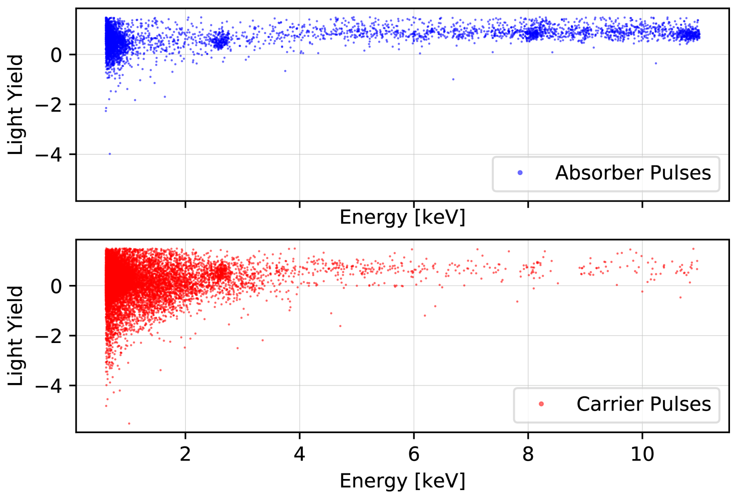


(b) Zoom view of the energy histogram.

Figure 7.15.: Energy histograms after the **High Cut** which is summarized in table 7.4. Absorber pulses are all pulses with a neural network output higher than the cut limit. Vice versa for carrier pulses. (a) Energy histogram of all pulses. (b) Zoom view into the region up to 11 keV which is approximately the maximum energy covered by the training data.



(a) Light yield plot.



(b) Zoom view of the light yield plot.

Figure 7.16.: Light yield plot after the **High Cut** which is summarized in table 7.4. Absorber pulses are all pulses with a neural network output higher than the cut limit. Vice versa for carrier pulses. (a) Light yield plot of all pulses. (b) Zoom view into the region up to 11 keV which is approximately the maximum energy covered by the training data.

8. Summary and Outlook

Neural Networks are a very versatile tool which can be used in many applications ranging from supervised, unsupervised as well as reinforcement learning. This work focuses on a supervised learning task, namely the discrimination between two different pulse classes of the TUM40 cryogenic detector used in CRESST. Many related applications are possible, such as pulse shape discrimination on similar detector designs, for example the ones used in the COSINUS experiment or discriminating between many classes contrary to two. Furthermore, only the phonon signal was used, the outcome of additionally considering the light signal should be investigated in the future. From great importance for the former mentioned tasks is the training data. In order to generalize features from the data those features have to be present in the training data. Therefore great emphasis was laid on the outlined artificial pulse generation method as it resembles the nature of real observed pulses.

As discussed the data consists of two different pulse shapes as well as an overlying parameter, which is the amplitude of the pulses. A CNN and a FNN/PNN are not capable of dealing with data of the same structure but with different scaling. This work dealt with this issue by creating a dataset which covered pulses of many different scales in order to have enough variety for the neural network to learn from. However, this is a rather rough approach as the needed variety of data leads to big datasets and long training. Another more sophisticated way of dealing with this circumstance should be investigated in the future.

The observed pulses show an additional, important feature. Namely, the pulses are likely to be shifted in time. It was observed experimentally that FNNs cannot deal with this form of data, hence the use of convolutional neural networks (CNNs) delivered an elegant way of dealing with this form of data.

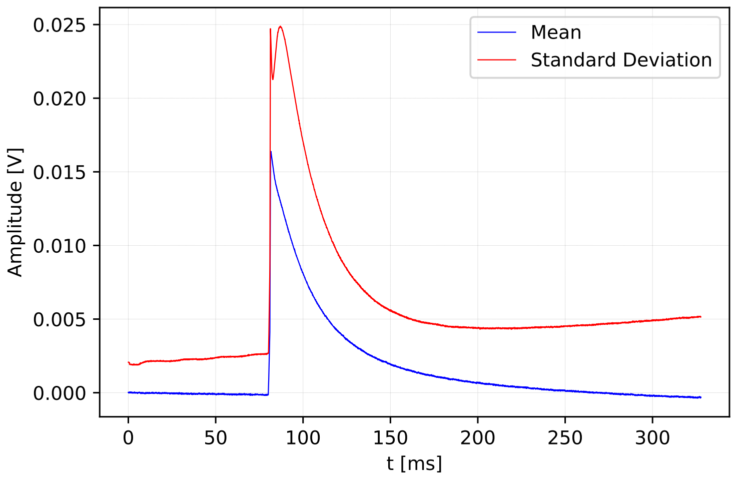
In section 3.2 former ways of pulse shape discrimination were discussed. In particular the method used in [26] which trained neural networks on the fitting parameters provided by the standard event fit was outlined. This present work skipped the standard event fitting procedure completely by training neural networks on the raw data. In summary a neural network model based on CNNs was found and applied to real data. The resulting statement is that indeed many observed low energy pulses are carrier events but also many absorber events are observed in the low energy region. Hence this work confirms the statement which was found in [26]. In contrast to work with data in real space, as the former methods did, discrimination in the Fourier space is a potential method. Two algorithms which are expected to work well together with data in the Fourier space are SVMs (Support Vector Machines) and the optimum filter method. Neural networks can be used as well in conjunction with data in Fourier space. These approaches should be investigated in the future.

Different application for neural networks in CRESST are imaginable. The introduced method of simulating noise with the power spectral density of measured empty baselines can be replaced by generative models based on neural networks. Promising algorithms are GANs (Generative Adversarial Networks) and Autoencoders. Those neural network algorithms are able to learn the distribution of the noise from empty baselines directly.

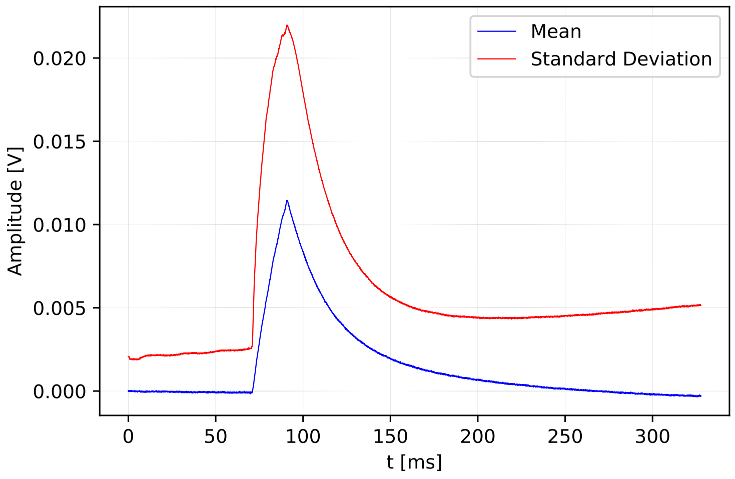
Another use-case is triggering events. In CRESST-III a continuous stream is saved and records are selected with the optimum filter technique described in [33]. CNNs or LSTMs could be trained on pulses such that triggering and energy reconstruction is done in one step.

A. Discussion of Data Pre-Processing

In section 5.2.2 data pre-processing was outlined to be a crucial step for machine learning algorithms. This is usually done by standardizing every feature according to equation (5.8). However, for the particular structure of TUM40 data this step can result in unwanted effects as follows. The noise ongoing present on the data can be roughly divided into a very fast and a very slow acting component. The slow acting component ranges over many records thus it is somewhat canceled as every record is shifted to zero by subtracting the mean of the first 200 samples from every sample. However, the fast noise component can roughly be modeled as white noise with a mean of approximately zero and a very small standard deviation («1). Thus standardizing data by equation (5.8) is numerically poor conditioned as small numbers are subtracted and small numbers are divided. Furthermore, this effect is amplified because the data is only provided in single precission floating point numbers. Figure A.1 illustrates the mean as well as the standard deviations per sample for the datasets with measured noise in order to show the order of magnitudes. Indeed skipping this data pre-processing step showed great improvement as the same model trained on the same data with and without data pre-processing was resulting in lower evaluation loss values by approximately one order of magnitude. It is mentioned that in all models batch normalization or self normalizing exponential unit activation functions were used. These concepts are discussed in section 5.3.3. It is likely that those methods provided a similar effect as the standardization given by equation (5.8) without the numerical drawbacks as they are meant to standardize the information flow through the neural network. In particular batch normalization uses the same procedure as seen in equation (5.8) but with an additional constant of 0.00001 added to the standard deviation present in the denominator for numerical stability. However, all this techniques are applied after every layer. Hence the input features still remain without transformation.



(a) Dataset with measured noise and non time-shifted pulses.



(b) Dataset with measured noise and time-shifted pulses.

Figure A.1.: Depiction of the mean as well as the standard deviation per sample for the datasets with measured noise. These quantities are derived over the training set consisting of 10000 pulses respectively.

B. Summary of Real Data

In order do derive meaningful pulses from all the observed data many cuts have to be applied. Hence some stages of the data preparation have to be carried out before neural networks can analyze the real data. A detailed outline of the used data preparation methods is given in [30]. This work analyzes data right before the carrier cuts. 89985 pulses are provided at that stage. Additionally, every pulse is fit with the absorber and the carrier template. Hence the energy reconstruction was carried out and the plots on the real data are depicted as a function of the reconstructed energy and not the pulse height parameter. Moreover the light yield parameter is given for every pulse. In principle negative numbers for the light yield parameter are nonphysical but are observed because in the low energy region the amplitude given by the standard event fit can be negative resulting in a negative light yield. Also very high nonphysical values far above 1.5 are observed and cut.

Bibliography

[1] Planck Collaboration. Esa homepage. Available at `http://www.esa.int/spaceinimages/Images/2013/03/Planck_cosmic_recipe` (07.01.2019).

[2] A. A. Penzias and R. W. Wilson. A Measurement of Excess Antenna Temperature at 4080 Mc/s. *Astrophysical Journal*, 142:419–421, 1965.

[3] D. J. Fixsen. The Temperature of the Cosmic Microwave Background. *The Astrophysical Journal*, 707:916–920, December 2009.

[4] E. Komatsu et al. Five-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Cosmological Interpretation. *Astrophys. J. Suppl.*, 180:330–376, 2009.

[5] P. A. R. Ade et al. Planck 2015 results. XIII. Cosmological parameters. *Astron. Astrophys.*, 594:A13, 2016.

[6] F. Zwicky. Die rotverschiebung von extragalaktischen nebeln. *Helvetica Physica Acta*, 6:110–127, 1933.

[7] Edvige Corbelli and Paolo Salucci. The Extended Rotation Curve and the Dark Matter Halo of M33. *Mon. Not. Roy. Astron. Soc.*, 311:441–447, 2000.

[8] Fabio Iocco, Miguel Pato, and Gianfranco Bertone. Evidence for dark matter in the inner Milky Way. *Nature Phys.*, 11:245–248, 2015.

[9] Douglas Clowe, Marusa Bradac, Anthony H. Gonzalez, Maxim Markevitch, Scott W. Randall, Christine Jones, and Dennis Zaritsky. A direct empirical proof of the existence of dark matter. *Astrophys. J.*, 648:L109–L113, 2006.

[10] Mordehai Milgrom. New Physics at Low Accelerations (MOND): an Alternative to Dark Matter. *AIP Conf. Proc.*, 1241:139–153, 2010.

[11] C. Alcock et al. The MACHO project: Microlensing results from 5.7 years of LMC observations. *Astrophys. J.*, 542:281–307, 2000.

[12] P. Tisserand et al. Limits on the Macho Content of the Galactic Halo from the EROS-2 Survey of the Magellanic Clouds. *Astron. Astrophys.*, 469:387–404, 2007.

[13] A. H. G. Peter. Dark Matter: A Brief Review. *ArXiv e-prints*, January 2012.

[14] Gianfranco Bertone, Dan Hooper, and Joseph Silk. Particle dark matter: Evidence, candidates and constraints. *Phys. Rept.*, 405:279–390, 2005.

[15] Spencer Chang, Ralph Edezhath, Jeffrey Hutchinson, and Markus Luty. Effective WIMPs. *Phys. Rev.*, D89(1):015011, 2014.

[16] Jens Michael Schmaler. *The CRESST Dark Matter Search – New Analysis Methods and Recent Results.* PhD thesis, Technische Universität München, 2010.

[17] G. Angloher et al. Exploring low-mass dark matter with cresst. *Journal of Low Temperature Physics*, 184(3):866–872, Aug 2016.

[18] G. Angloher et al. Commissioning Run of the CRESST-II Dark Matter Search. *Astropart. Phys.*, 31:270–276, 2009.

[19] Andrew Brown, Sam Henry, Hans Kraus, and Christopher McCabe. Extending the CRESST-II commissioning run limits to lower masses. *Phys. Rev.*, D85:021301, 2012.

[20] G. Angloher et al. Results from 730 kg days of the cresst-ii dark matter search. *The European Physical Journal C*, 72(4):1971, Apr 2012.

[21] G. Angloher et al. Results on low mass WIMPs using an upgraded CRESST-II detector. *Eur. Phys. J.*, C74(12):3184, 2014.

[22] G. Angloher et al. Results on light dark matter particles with a low-threshold CRESST-II detector. *Eur. Phys. J.*, C76(1):25, 2016.

[23] G. Angloher et al. Probing low WIMP masses with the next generation of CRESST detector, 2015.

[24] F. Petricca et al. First results on low-mass dark matter from the CRESST-III experiment. In *15th International Conference on Topics in Astroparticle and Underground Physics (TAUP 2017) Sudbury, Ontario, Canada, July 24-28, 2017*, 2017.

[25] M. Ambrosio et al. Vertical muon intensity measured with macro at the gran sasso laboratory. *Phys. Rev. D*, 52:3793–3802, Oct 1995.

[26] Andreas Josef Zöller. *Artificial Neural Network Based Pulse-Shape Analysis for Cryogenic Detectors Operated in CRESST-II.* PhD thesis, Technische Universität München, 2016.

[27] F. Reindl et al. The CRESST Dark Matter Search - Status and Perspectives. In *Proceedings, 12th Conference on the Intersections of Particle and Nuclear Physics (CIPANP 2015): Vail, Colorado, USA, May 19-24, 2015*, 2015.

[28] CRESST Collaboration. Cresst - detector concepts. Available at `https://www.cresst.de/` (09.11.2018).

[29] F Pröbst, Matthias Frank, S Cooper, P Colling, D Dummer, Paul Ferger, G Forster, A Nucciotti, W Seidel, and L Stodolsky. Model for cryogenic particle detectors with superconducting phase transition thermometers. *Journal of Low Temperature Physics*, 100:69–104, 07 1995.

[30] Florian Reindl. *Exploring Light Dark Matter With CRESST-II Low-Threshold Detectors.* PhD thesis, Technische Universität München, 2016.

[31] Michael Kiefer. *Improving the Light Channel of the CRESST-II Dark Matter Detectors.* PhD thesis, Technische Universität München, 2012.

[32] R. Strauss et al. A detector module with highly efficient surface-alpha event rejection operated in CRESST-II Phase 2. *Eur. Phys. J.*, C75(8):352, 2015.

[33] S. Di Domizio, F. Orio, and M. Vignati. Lowering the energy threshold of large-mass bolometric detectors. *JINST*, 6:P02007, 2011.

[34] D. Dal Bosco. Noise analysis in the cresst experiment. Master's thesis, Eberhard Karls Universität Tübingen, 2018.

[35] M. Carrettoni and O. Cremonesi. Generation of noise time series with arbitrary power spectrum. *Computer Physics Communications*, 181:1982–1985, December 2010.

[36] Martin Stahlberg. Phd thesis. Unpublished, In Preparation.

[37] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

[38] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[40] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147. JMLR.org, 2013.

[41] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[42] Computer Science Department Stanford University. Cs231n: Convolutional neural networks for visual recognition. Available at `http://cs231n.stanford.edu/` (09.11.2018).

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[44] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[45] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017.

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[47] Lutz Prechelt. Early stopping - but when? In *Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2*, pages 55–69. Springer-Verlag, 1997.

[48] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[49] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560, 2016.

[50] Pierre Baldi, Kyle Cranmer, Taylor Faucett, Peter Sadowski, and Daniel Whiteson. Parameterized neural networks for high-energy physics. *Eur. Phys. J.*, C76(5):235, 2016.

[51] D. M. W. Powers. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[52] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[53] PyTorch. Pytorch homepage. Available at `https://pytorch.org/` (08.01.2019).

[54] PyTorch. Pytorch documentation. Available at `https://pytorch.org/docs/stable/index.html` (08.01.2019).