The way to a smarter community: Exploring and Exploiting Data Modeling, Big Data Analytics, High-Performance Computing and Artificial Intelligence Techniques for Applications of 2D Energy-Dispersive Detectors in the Crystallography Community.

DISSERTATION
zur Erlangung des Grades eines Doktors
der Naturwissenschaften

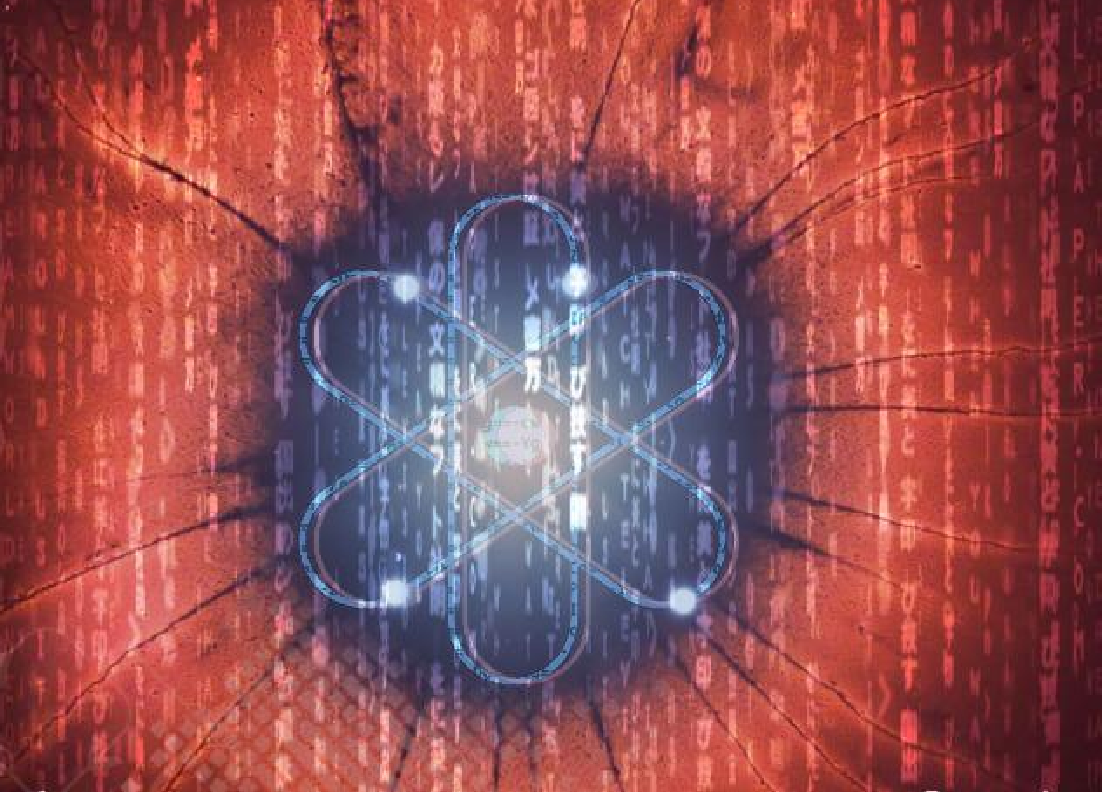
vorgelegt von
M. Sc. Amir Tosson

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen
Siegen 2020

University of Siegen, Germany

Doctoral Thesis

The way to a smarter community: Exploring and Exploiting Data Modeling, Big Data Analytics, High-Performance Computing and Artificial intelligence Techniques for Applications of 2D Energy-Dispersive Detectors in the Crystallography Community.



Author
Amir Tosson

Supervisor
Prof. Dr. Ullrich Pietsch

A thesis submitted in fulfillment of requirements for the degree of Doctor rerum naturalium (Dr. rer. nat.)

in the

Solid state physics
Department of Physics

Betreuer und erster Gutachter
Prof. Dr. Dr. h.c. Ullrich Pietsch
Universität Siegen

Zweiter Gutachter
Prof. Dr. Jonathan Taylor
Universität Kopenhagen

Tag der mündlichen Prüfung
23. June 2020

UNIVERSITY OF SIEGEN, GERMANY

Abstract

Faculty IV
Department of Physics

Doctor rerum naturalium (Dr. rer. nat.),

***The way to a smarter community:* Exploring and Exploiting Data Modeling, Big Data Analytics, High-Performance Computing and Artificial Intelligence Techniques for Applications of 2D Energy-Dispersive Detectors in the Crystallography Community.**

by Amir TOSSON

Big data will be a source of new economic value and innovation. But even more is at stake. Big data's ascendancy represents [...] shifts in the way we analyze information that transforms how we understand and organize society" [1]. This important statement by Mayer-Schönberger and Cukier highlights the crucial benefit this field offers to the research and business communities. It is opening up an entirely new horizon for economic and innovative solutions. It also sheds light on the main challenge researchers and business analytics across the world have been facing during the recent years, namely big data handling, extraction and employing. Digitization enabling the real-time data visualization for the further analysis is now a vital prerequisite for a successful and innovative scientific community. This work aims to contribute to the transformation of the crystallography community, dealing with an enormous volume of the measured data obtained due to technological advancement in radiation and detection, into a smarter scientific environment.
Specifically, this dissertation has a twofold aim. First, it is intended to provide an introduction and implementation guideline to scientists in the crystallography community and across the fields demonstrating how trend technologies might be applied to design cutting-edge research projects. Second, considering the Energy Dispersive Laue Diffraction (EDLD) as a case study, it introduces an innovative approach to exploit high-performance computing to develop a novel framework for data processing within the time frame of a few seconds compared to the traditional analytic systems requiring a few hours to process the same amount of data. The framework developed is based on multiple artificial intelligence algorithms, designed to perform tasks that cannot be processed by basic programming techniques. Extending this approach, data clouding has been employed to establish the communication channels between scientists and the collected data. This computing solution helps to achieve commoditization of computational resources, implementation of open-source software, data virtualization, globalization of workforce, establishing a data-sharing point. On the whole, due to these technological improvements, the crystallography community might gain maximum benefit from the collected data. As a proof of concept, the reliability, efficiency, and performance of the entire work has been verified by involving the system in a challenging task, namely: the one-shot analysis of the micro texture in polycrystalline materials.

UNIVERSTÄT SIEGEN, DEUTSCHLAND

Zusammenfassung

Der Weg zu einer intelligenteren Gemeinschaft: Erforschung und Nutzung von Datenmodellierung, Großdatenanalyse, Hochleistungsrechner und Techniken der künstlichen Intelligenz für die Anwendung von 2D-Energiedispersiven Detektoren in der Kristallographie-Gemeinschaft.

von Amir TOSSON

Die Verarbeitung großer Datenmengen bietet eine immer größeren Wert für Wirtschaft und neue Innovationen. Aber es bietet noch viel größeres Potenzial. Die wachsende Menge an Daten spiegelt [...] Veränderungen in der Art und Weise, wie wir Informationen analysieren, die unser Verständnis und unsere Organisation der Gesellschaft verändern" [1], wieder. Diese wichtige Aussage von Mayer-Schönberger und Cukier unterstreicht den entscheidenden Nutzen, den dieses Feld für die Forschung und die Wirtschaft bietet. Es eröffnet einen völlig neuen Horizont für wirtschaftliche und innovative Lösungen. Sie beleuchtet auch die größte Herausforderung, vor der Forscher und Wirtschaftsanalytiker in den letzten Jahren weltweit stehen, nämlich die Handhabung, Extraktion und Nutzung großer Datenmengen. Die Digitalisierung, die die Echtzeit-Visualisierung der Daten für die weitere Analyse ermöglicht, ist heute eine entscheidende Voraussetzung für eine erfolgreiche und innovative wissenschaftliche Gesellschaft. Diese Arbeit soll dazu beitragen, die KristallographieGesellschaft, die mit einer enormen Menge an Messdaten, die aufgrund des technologischen Fortschritts in der Strahlung und Detektion gewonnen wurden, in eine intelligentere wissenschaftliche Umgebung zu verwandeln.

Diese Dissertation verfolgt insbesondere ein zweifaches Ziel. Erstens soll sie eine Einführung und einen Umsetzungsleitfaden für Wissenschaftler in der KristallographieGemeinschaft und in allen Bereichen bieten und aufzeigen, wie Trendtechnologien zur Gestaltung von Spitzenforschungsprojekten eingesetzt werden können. Zweitens wird unter Berücksichtigung der energiedispersiven Laue-Beugung (EDLD) als Fallstudie ein innovativer Ansatz zur Nutzung der Hochleistungsrechnen vorgestellt, um ein neuartiges Rahmenwerk für die Datenverarbeitung innerhalb des Zeitraums von wenigen Sekunden zu entwickeln, im Vergleich zu den traditionellen analytischen Systemen, die für die Verarbeitung der gleichen Datenmenge einige Stunden benötigen. Das entwickelte Rahmenwerk basiert auf mehreren Algorithmen der künstlichen Intelligenz, die so konzipiert sind, dass sie Aufgaben ausführen können, die durch grundlegende Programmiertechniken nicht verarbeitet werden können. In Erweiterung dieses Ansatzes wurde die Data Clouding eingesetzt, um die Kommunikationskanäle zwischen den Wissenschaftlern und den gesammelten Daten her- zustellen. Diese Rechenlösung hilft bei der Kommodifizierung der Rechenressourcen, der Implementierung von Open-Source-Software, der Datenvirtualisierung, der Globalisierung der Arbeitskräfte und der Einrichtung eines Datenaustauschpunktes. Insgesamt könnte die Kristallographie-Gemeinschaft aufgrund dieser technologischen Verbesserungen maximalen Nutzen aus den gesammelten Daten ziehen. Als Machbarkeitsnachweis wurde die Zuverlässigkeit, Effizienz und Leistung der gesamten Arbeit durch die Einbeziehung des Systems in eine anspruchsvolle Aufgabe verifiziert, nämlich: die einmalige Analyse der Mikrotextur in polykristallinen Materialien.

List of publications:

1. Tosson, A, Shokr, M, Abboud, A, Algashi, A, Hartmann, R, Strüder, L, & Pietsch, U. EDLD-Tool: A real-time GPU-based tool to stream and analyze energy-dispersive Laue diffraction of BIG Data sets collected by a pnCCD. Journal of Instrumentation 14, P01008 (2019).
2. Tosson, A, Bahrami, D, Davtyan, A, Shokr, M & Pietsch, U. Deep learning application for events classification of energy-dispersive Laue diffraction datasets collected by pnCCD. International Journal of Modern Engineering Research (2019).
3. Tosson, A, Shokr, M & Pietsch, U. Application of cloud computing for big data in the X-ray crystallography community. The 3rd International Conference on Software Engineering and Information Management (2020) Sydney, Australia.
4. Tosson, A, Al Humaidy, M, Shokr, M & Pietsch, U. Application of ML in grain-related classification of Laue spots in a polycrystalline ED Laue pattern. (In progress).
5. Shokr, M, Tosson, A , Abboud, A, & Pietsch, U. Energy-dispersive Laue diffraction by means of a pnCCD detector coupled to a CsI(Tl) scintillator using ultra-hard X-ray synchrotron radiation. Journal of Synchrotron Radiation (2019).
6. Shokr, M, Schlosser, D, Abboud, A, Algashi, A, Tosson, A , Conka, T, Hartmann, R, Klaus, M, Genzel, C, Strüder, L, & Pietsch, U. Applications of a pnCCD detector coupled to columnar structure CsI(Tl) scintillator system in ultra high energy X-ray Laue diffraction. Journal of Instrumentation (2017).
7. Abboud, A, Dönges, B, Shokr, M, Tosson, A , Micha, JS , Hartmann, R, Strüder, L, & Pietsch, U. ELattice tilt and subdivision of grains during crack formation in VHCF duplex stainless steel using microbeam x-ray laue diffraction. VHCF7 Seventh International Conference on Very High Cycle Fatigue (2017), Dresden , Germany.
8. Poster: Tosson, A, Shokr, M & Pietsch, U. Application of Machine Learning in the Energy Dispersive Laue Diffraction experiments. Artificial Intelligence Applied to Photon and Neutron Science workshop, Grenoble, France.
9. Poster: Tosson, A, Shokr, M & Pietsch, U. Employing Big Data Analytics, High-Performance Computing, and Artificial Intelligence for application of 2D Energy-Dispersive detectors. RACIRI summer school, Kaliningrad, Russia.
10. Poster: Tosson, A, Abboud, A, Shokr, M, Hartmann, R, Strüder, L, & Pietsch, U. Parallel Computing for Analysis of Big Data in Solid State physics. HiPEAC conference, Manchester, UK.

Contents

List of Figures

List of Tables

List of Abbreviations

FAIR	**F**indability, **A**ccessibility, **I**nteroperability **R**euse
HPC	**H**igh **P**erformance **C**omputing
EDLD	**E**nergy **D**ispersive **L**aue **D**iffraction
AI	**A**rtificial **I**ntelligence
MiTx	**Mi**cro **T**e**x**ture
XRD	**X**-**R**ay **D**iffraction
FEL	**F**ree **E**lectron **L**aser
CCD	**C**harge **C**oupled **D**evices
ADC	**A**nalog-to-**D**igital **C**onverter
NDT	**N**on-**D**estructive **T**est
API	**A**pplication **P**rogramming **I**nterface
EDA	**E**xploratory **D**ata **A**nalysis
IMDB	**I**n-**M**emory **D**ata**b**ase
SaaS	**S**oftware **as a S**ervice
PaaS	**P**latform **as a S**ervice
IaaS	**I**nfrastructure **as a S**ervice
DW	**D**ata **W**arehouse
CDW	**C**loud-based **D**ata **W**arehousing
DOI	**D**igital **O**bject **I**dentifier
MVVM	**M**odel **V**iew **V**iew**M**odel
CLI	**C**ommon **L**anguage **I**nfrastructure
CPU	**C**entral **P**rocessing **U**nit
GPU	**G**raphics **P**rocessing **U**nit
ML	**M**achine **L**earning
DL	**D**eep **L**earning
HC	**H**ierarchical **C**lustering
SSE	**S**um of **S**quared **E**rrors
NN	**N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
ReLU	**Re**ctified **L**inear **U**nit
YOLO	"**Y**ou **O**nly **L**ook **O**nce

Chapter 1

Introduction

There is an undoubted fact: the next decade will witness a massive increase in the volume of the collected data. Nowadays, data are analogous to gold, and data management is considered as a critical differentiator for market winners. Moreover, the pace of development of any business depends on the systematic harvest of data. According to a study by the World Economics Forum [2], it is estimated that by 2020 the globally generated data will reach 44 zettabytes. That is the equivalent to one thousand million hard-disks with the storage capacity of one terabyte!. All these facts force the decision-makers to seek new innovative approaches for data handling endeavoring maximum benefit for the respective business.
The modern scientific society seems to experience a drastic boom in the nature of its data. It has gained an unprecedented ability to generate and store breathtaking amounts of data. For example, particle accelerators (e.g. the Large Hadron Collider at CERN) generate $\approx$ 100 terabytes of data per day [3]. In addition, the fourth generation high brilliance X-ray sources combined with 2D detectors have increased the rate of data generation up to tens of GByte/sec [4]. This society tries to evolve the ability to leverage the data in such a way as to distill from them useful knowledge, hence providing new discoveries.

1.1 Crystallography community

Crystallography community brings together scientists that are active in many fields (e.g., X-ray diffraction, neutron scattering, electron microscopy, etc.) in order to enhance experimental experience, answer scientific questions, and introduce new discoveries. It is the only available tool to dive deeply into the micro- and nanosystems of this universe. Crystallographers may use different radiation sources to characterize the inner structures of crystals to identify their behavior under different conditions. This generates precious data and knowledge that can be utilized by many other scientific branches, such as chemistry, physics, and biology.
This field was born in 1901 when Röntgen was awarded the Nobel Prize for X-ray discovery [5]. During the following few years, Max von Laue established his equation that investigates the interaction of X-rays with crystals generating a diffraction pattern or a Laue pattern. In 1913 Bragg underpinned the crystallography by introducing his famous formula, known today as *"Bragg's low"* [6]. He formulated the relationship between atomic structure of any crystal and its generated X-ray diffraction pattern. This could support the community with a tool to enhance the knowledge about the structure of matter (i.e. materials, catalysts, DNA, proteins, and viruses).
Nowadays, any progress in crystallography is growing out of the development of the primarily two components of any successful experiment, namely: the radiation

source and the detector. Thanks to the new technologies in industry and manufacturing, many synchrotron radiation facilities are available providing a high brilliance X-ray beam. Furthermore, the new generation of fourth dimensional detectors has opened the door for further discoveries beyond the expectations. These factors and others have caused a bottlenecking situation: a congested data flow system. It often happens when the data generating rate overrides the data managing rate. Such an issue may show several deficiencies, such as losing either valuable data or value of the generated data. Therefore, many efforts have been done to promote the standards of digital data.

1.2 FAIR data concept

"FAIR Guiding Principles for scientific data management and stewardship" [7] are a set of principles providing guidelines to improve the **F**indability, **A**ccessibility, **I**nteroperability, and **R**euse of digital assets, maximizing the integrity and impact of the research investment. They address an effective approach to promote maximum use of research data, enhancing the ability of machines to automatically find and use the data. They emphasize that meta-data and data should be easy to be found, assigned by a global and persistent identifier, authorized and authenticated accessible, integrated with other data, and according to community policies.

1.3 The vision

As a part of the scientific community, the Solid state physics department of University of Siegen has framed a novel concept to digitalize and automize the whole data processing pipeline. Starting from data acquisition to data analysis, it is performed with the possible minimum user intervention. Aiming to achieve a suitable scientific environment that provides computational resources commoditization, open-source software implementation, virtualization, workforce globalization, and to establish a data-sharing point. This enables the community to extract the maximum benefit of the collected data achieving the FAIR data concept. Moreover, it can expedite the time to discovery, allowing the scientists to focus on their science without to be experts in data management. Therefore, many resources have been committed to serve not only for the final product but also training the human cadres who can connect the science with the different fields.
Deeming this vision, this work brings different computing technologies to the scientific community. It aims to provide the scientists with guidelines about each trend technology, helping them to extent their understanding of the different techniques. It also demonstrates a practical example to implement these technologies in the crystallography community. Considering the Energy Dispersive Laue Diffraction (EDLD) experiment as a case of study, a new framework has been gradually developed from scratch.

1.4 Structure

This work is organized as follows: Chapter 2 introduces the fundamentals of crystallography and detector technologies, followed by a detailed investigation of the experimental approach and its concerns, which used as a case of study in this work. Chapter 3 deals with the Big Data and its basics. It addresses the implementation of Big Data strategies (e.g. data warehousing and clouding) in the X-ray community. Chapter 4 is devoted to the investigation of High-Performance Computing (HPC) essentials and applications. It includes a description of the developed software package to achieve the full real-time analysis, namely: *EDLD-tool*. On the basis of Artificial Intelligence and its applications, chapter 5 demonstrates a preparatory introduction of AI techniques, architectures, and usage. Then, it focuses on the procedures to exploit such algorithms. Based upon that, a verification of the eligibility and worthiness of the developed software to extend the application range of the studied experimental approach is presented in chapter 6. The focus of this verification study is on material Micro-Texture (MiTx) analysis using a *one-shot* experiment, which has been a challenging task to be achieved because of the complicated structure of its datasets. This work concludes, in chapter 7, with a description of the achievements, the limitations of this study, and an outlook for future research. It is worth to mention that this work is based on recent publications [8–10].

Chapter 2

X-ray Diffraction theoretical framework and empirical concerns

X-ray Diffraction (XRD) is one of the most powerful non-destructive tools to characterize different types of matter, such as fluids, powders, and crystals. It is able to provide detailed information about the crystallographic structure, chemical composition, and physical properties of materials [11]. It is based on the elastic scattering of X-ray photons by atoms that are arranged in a periodic lattice, generating scattering X-rays that are in phase and produce constructive interference. This is explained by the means of Bragg's law where the wavelength of electromagnetic radiation is measured by the diffraction angle and the lattice spacing in a crystalline sample, as follows:

$$n\lambda = 2d.\sin(\theta) \tag{2.1}$$

where n is an integer called the order of reflection, λ is the wavelength of X-rays, d is the interplanar spacing between the atomic planes of a given material and θ is called the diffraction angle and represents the angle between the incident beam and the surface to the probed crystallographic plane. Figure 2.1 shows the interaction of X-rays with the atoms in a crystal. It displays two incident rays on two atomic layers separated by the distance d. Figure 2.1a visualizes constructively interfering rays, satisfying Bragg's law. On the other hand, figure 2.1b represents the two rays with a phase shift that causes a destructive interference. The collection of many of these rays results in, so-called, diffraction patterns. Analyzing such patterns can provide a detailed structural characterization of a material, such as composition and molecular connectivity, crystal size and orientation, stress and strain determination, and others [12, 13]. Currently, XRD is used in wide range of medical and material research areas.
In order to achieve a successful XRD experiment, three main aspects should be satisfied, namely: a source, a detector, and an analysis procedure. Any advancement in XRD depends on what is the range of energy and the brilliance provided by the X-ray source, how much efficiency and sensitivity can be achieved by the detector, and how fast and accurate is the analysis procedure. This chapter represents theoretical framework and empirical concerns about the components of XRD. It is organized so that, section 2.1 covers the different sources of X-rays and section 2.2 is a collective summary of the detection technologies. Section 2.3 is devoted to explain the EDLD experiment as an analysis procedure giving a brief about its experimental geometries and analytical pipeline.

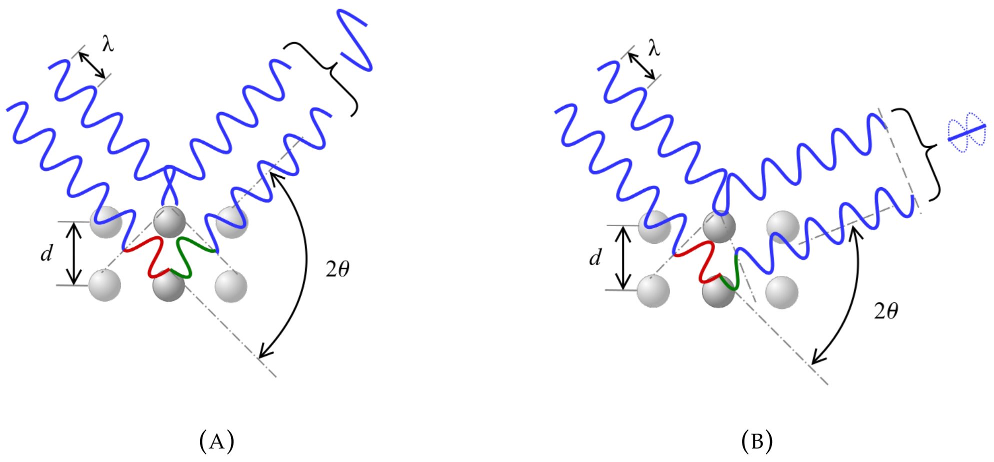


FIGURE 2.1: The diffraction of x-rays by crystal planes according to Bragg's law. (A) is a constructive interference, while (B) is a destructive interference (source [14]).

2.1 X-rays sources

X-rays are electromagnetic waves that have frequencies in the range 3×10^{16} Hz to 3×10^{19} Hz, corresponding to wavelength of 0.01 to 10Å . They have particle-like properties and can be considered as bunches of photons carrying discrete amounts of energies in the range 100 eV to 100 keV. This is known as the dual (wave-particle) nature of X-rays [15]. There are three common mechanisms used to produce X-rays, namely; conventional X-rays tubes, synchrotron radiation, and Free Electron Laser (FEL).

2.1.1 Conventional X-rays tubes

A typical X-rays tube for crystallographic studies is to hit the target or the anode by an energetic beam of accelerated electrons that are usually produced by heating a metal filament [16, 17]. The energy of the produced X-ray photons can have a value up to the maximum energy of the incident electrons. This process is defined by two different atomic processes:

2.1.1.1 Bremsstrahlung

The name bremsstrahlung is a german name means "braking radiation". It is electromagnetic radiation given off by the acceleration or deceleration of a charged particle (e.g. an electron or an positron) [16, 18]. It occurs when an ultra relativistic particle (i.e. it has velocity close to the speed of light) interacts with a strong electric or magnetic field. Such an interaction results in a continuous distribution of X-rays radiation.

2.1.1.2 Characteristic X-rays

If the electron has enough energy, it can knock an orbital electron out of the inner electron shell of a metal atom. Since the process leaves a vacancy in the electron energy level from which the electron came, the outer electrons of the atom cascade down to fill the lower atomic levels, and one or more characteristic X-rays are usually emitted. As a result, sharp intensity peaks appear in the spectrum at wavelengths that are a characteristic of the material from which the anode target is made. The frequencies of the characteristic X-rays can be predicted from the Bohr model.

2.1.2 Synchrotron radiation

Synchrotron accelerators were originally developed for particle physics. From 1990, facilities have been elected especially for X-ray use to overcome the disadvantage of the conventional X-ray source: the constant compensation of the energy losses during interacting. When an electron or a positron emits a photon, it slows down and its trajectory is deflected. Thus, synchrotron accelerators are designed to provide abundant bremsstrahlung photons which called "*synchrotron radiation*" [19]. Such accelerators are widely preferred because of their high brilliance and level of polarization of the generated beam. Moreover, the resulting X-rays are energy-tunable.
A typical synchrotron consists of:

- LINAC which is a device similar to the cathode ray tubes to produce the electrons for the storage ring. These electrons are packed in "bunches" and then accelerated enough for injection into the booster synchrotron.
- Booster ring where the electrons are accelerated before being injected into the storage ring. The booster works to only when the storage ring is refilled.
- Storage ring where the electrons accelerate close to the speed of light. It is a tube maintained at very low pressure. As the electrons travel round the ring, they pass through different types of magnets and in the process produce X-rays.
- Beamlines are the final destination of the X-ray beams that surround the storage ring in the experimental hall. Each beamline is designed for use with a specific technique or for a specific type of research.

2.1.3 Free Electron Laser (FEL)

FELs are a noval type of laser where the optical amplification is achieved by accelerating high energetic (relativistic) particles to generate the beam which propagates through a periodic magnetic field (an undulator) [20]. A FEL directly converts the kinetic energy of the electron beam into light with high tunability and potentially high power and efficiency. This can create a much higher spectral brightness and linear coherent beam, compared with other synchrotron radiation sources [21]. Free electron lasers offer a number of valuable attractions, such as the ability to be operated in very wide wavelength ranges and the spectacular performance in extreme wavelength region that are not reachable by any other light sources. This can be used in many applications such as atomic and molecular physics, ultrafast X-ray science, advanced material studies, biology and medicine.

2.1.4 Buzzwords in X-ray

- Soft X-rays: are the rays with photon energies below $5 - 10$ keV. They are easily absorbed and can only be used by evacuated beamlines.
- Hard X-rays: are the rays with high photon energies above $5 - 10$ keV. They have a strong penetrating ability and are widely used in imagining inside of visually opaque objects. The hard X-rays can be used outside vacuum.

- Monochromatic beam: is X-rays of a fixed wavelength (i.e. discrete energy spectrum). It can be produced by characteristic radiation emitted by a conventional X-ray tube, monochromatized synchrotron radiation, or free electron lasers [22].
- White beam: is X-rays of a ranged wavelength (i.e. continuous energy spectrum). It can be produced by bremsstrahlung delivered by an X-ray tube or white synchrotron radiation [22].

2.2 Detector technology

Many sophisticated applications such as medical imaging, industrial inspection, and High Energy Physics (HEP) research require detectors with special characteristics, such as fast frame rate, high spatial resolution, dynamic energy range, and convenient active sensor area. Therefore, there is no exaggeration to say that the radiation detectors are the key element of any scientific achievement. The rapid development of the X-rays sources technology motivates the scientific community to push forward the detector development beyond its limits. The main aim is to design photon detectors that can deal effectively with the extreme flux generated by the current and new generations of synchrotron and FEL sources. Thanks to the evolution in microelectronics, many new types of detectors are commercially available where each photon or particle can be treated individually, achieving the required spectral resolution, spatial resolution, and quantum efficiency. This part gives a short summary about the different types of available detectors, focusing on the 2D energy-dispersive ones.

2.2.1 Point detectors

From their name, point detectors are capable to discriminate X-ray photons in one specific spatial point. They can be categorized into two groups depending on the number of measured dimensions[1]. 1) 1D point detectors (i.e. proportional and scintillation counters) [23]: they retrieve the intensity information within a specific point, often called *"the observation point"*. The typical count rate of such technology is up to 10^5 counts per second [22]. It is essential to mention that, linear or curved point provide one-dimensional position resolution [24]. However, the disadvantages of point detectors are their large dead times and poor energy resolution. This may limit their applications specially in case of white X-rays. 2) 2D point detectors (i.e. energy-dispersive point detectors) [25, 26]: they are special types of point detectors based on the fully depleted pn-structures. They deliver the scattered X-ray spectrum at the detection point as an extra dimension besides the intensity information. They show some unique advantages in the case of in-situ Energy-Dispersive (ED) experiments using the white X-rays beam, compared with the conventional point detectors.

2.2.2 Area detectors

Area detectors are a class of X-ray detectors that are formed by two-dimensional structures of individual active units. They are used to deliver the intensity of an incident X-ray flux along one or more dimensions, such as position, time, or energy. Generally, they convert the incident X-ray photons into electrical signals which are detected by the pixel electronics. Pixel signals are then sent to further shared circuitry, which is accessible by all other detection channels.

[1]In this work, the intensity is considered as a dimension. Many literatures exclude the intensity.

One of the common X-rays area-detector is the Imaging Plate (IP). Originally, it was developed in Japan by Fuji Photo Film Co. Ltd for diagnostic radiography applications [27]. The system makes use of the photostimulable Eu-doped phosphor powder on plastic sheet that has a good capability to store information generated by X-rays [28, 29]. The incident X-rays photon excite the Eu^{+2} to Eu^{+3} which is, then, deexcited by a readout red laser. This results in losing an electron accompanied by emission of light with an intensity proportional to the incident X-ray flux[27]. IP are relatively cheap and reusable. They are efficient in delivering position information as they can monitor a 2D spatial resolution with an overall resolution of 3D (including the intensity). However, they show inability to deliver energy information and no more available.
Another semiconductor area detector is the Charge-Coupled Devices (CCD). They are considered to be the digital version of the area detectors with an extra dimension, namely; time resolution with an overall resolution of 4D. The CCDs are filmless and can capture light converting it to digital data. They are based on creating electron-hole pairs by the incoming X-ray photons which are typically realized in a multiple array of silicon wafer. The pixels signals are then converted into digital values by an Analog-to-Digital Converter (ADC). This can be done by determining the amount of generated charge at each photosite and converting that measurement to binary form. Currently, the Pilatus [30] and Eiger [31] detectors are the most efficient area detectors as they can reach a count rate above 10^6 count per pixel. Despite they show an excellent performance in the case of monochromatic applications, they failed to provide a sufficient energy resolution to be utilized in the white-beam mode.

2.2.3 2D energy-dispersive detectors (pnCCD)

The 2D energy dispersive detectors are special type of the CCD developed for time resolved, simultaneous position and energy dispersive detection of X-rays with low noise level, fast readout, and high quantum efficiency [32]. The pnCCD type is one of the most common used devices. The "pn" refers to the semiconductor pn-junctions that fabricated from both P-type and N-type regions and operate as reverse-biased diodes. Originally, it was developed at the MPI-Halbleiterlabor (MPI-HLL) to be used in X-ray astronomy application [33]. In 2007, the MPI-HLL and the University of Siegen have initialized a collaboration to exploit the advantages of the pnCCDs for X-ray crystallography applications [34]. The concept of the pnCCD as an energy-resolving area detector is based on the principle of sideward depletion in high resistivity silicon. The system serves as flexible large area detector in order to both resolving single photons in the spectroscopic operation mode and counting photons with a high dynamic range in the single photons counting mode. Moreover, the long-term stability and high radiation hardness are the pioneering benefits of such systems. The pnCCD used by the Siegen group has an entrance window covers area of 8.3 cm^2 with 384 $\times$ 384 pixels, where each pixel has a size of 75 $\times$ 75 μm^2. A fully depleted high-resistivity silicon volume of $450\mu m$ thickness creates a contact between the entrance window and a thin n-doped epitaxial layer with thickness of $7\mu m$. It is equipped with a new frame store module which allows for a readout frequency up to $1kHz$ with acquisition rate $\approx 112MB/s$ [35, 36]. The pnCCD delivers datasets that have a 4D structure (i.e. 2D position information, energy and the time resolution). This result in a 5D overall resolution, including the intensity [8]. It worth mentioning that the pnCCD has two different operation modes [22]: (1) The Single Photon Counting mode (SPC) mode, individual photons are resolved within a 3D data volume [22]. Charge clouds generated by photons should be separated in space and time by the means of sufficiently low count rate or by analytical techniques. (2) The integration mode, in which the pnCCD operates as a position resolving CCD.

2.3 Energy-Dispersive Laue Diffraction (EDLD)

Energy-dispersive Laue diffraction (EDLD) is one of the experiments which profits most substantially from the two dimensional energy-dispersive detectors (i.e pnCCD cameras) properties and high brilliance X-ray sources. It is a a Non-Destructive Test (NDT) tool to investigate the physical and mechanical characteristics of all kinds of materials. It is a convenient technology that requires a little or no sample preparation. EDLD is based on Brag's law and Laue equation, whereby both the angular positions and the diffracting energies of the Laue spots can be analyzed without additional information, allowing for a quick identification and quantification of the crystal structure and the respective lattice parameters. The one-shot nature, low cost, and rapid elemental analysis make EDLD an attractive and reliable tool for many applications. Moreover, the EDLD gives the chance to utilize an effective and systemic data-handling pipeline. This can synthesize and leverage the experimental datasets to enhance the overall goal of achievement.

2.3.1 Experimental setup

The primary advantage of EDLD experimental geometry is its simplicity and feasibility. Since the EDLD is an angle- and energy-dispersive tool, it is possible to avoid the prevalent experimental complexities, such as sample rotation and beam alignment. Therefore, it does not require any special devices or extraordinary experiment preparation steps. Figure 2.2 shows the typical experimental setup of the EDLD experiments. The used spectrum of white X-rays radiation is usually provided by the storage rings or conventional X-ray tubes and can range from 5 keV up to about 140 keV. A collimating system is used to tailor the beam size, if required. In most cases, an absorber system has to be installed in order to reduce the X-ray flux, allowing to run the detection system in the single photon counting mode [37]. The sample is located at a certain distance in front of the detector for transmission geometry, or behind for reflection geometry. Both the sample and the detector may be equipped by high resolution motor stages with steps sizes down to $1\mu m$, enabling to change the scattering geometry.
During the exposure time, the incident beam scatters elastically from the sample and the diffracted signals are collected by the detector. Due to the wide energy range of the attenuated beam, a characteristic multi-reflections Laue pattern is formed in addition to the X-ray fluorescence signal originated by the elements which are presented in a sample and experimental equipments. This pattern contains intense reflections, often called Laue spots, representing each a crystal plane which fulfills the Bragg condition. However, usual 2D detectors are not able to identify and analyze the collected Laue spots without preliminary information about the experiment condition (e.g. incident beam energy, material lattice parameters, or sample orientation).

Therefore, to achieve the maximum benefit of the EDLD, a 2D energy-dispersive detection system is often used. In case of energy values higher than 80 kev, it is recommended to use the bare silicon chip attached to a scintillator (e.g CsI:Tl scintillator) to realize a suitable quantum efficiency [37].

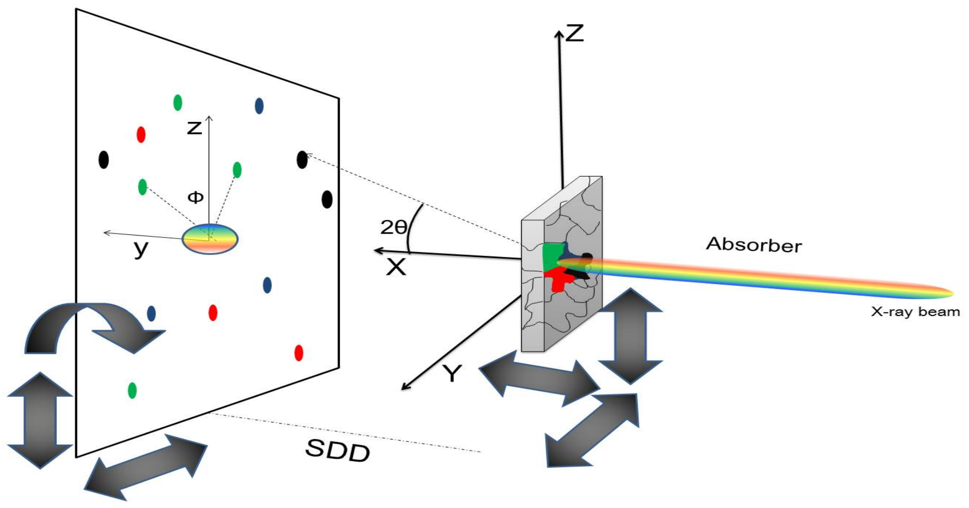


FIGURE 2.2: The EDLD experimental setup. X, Y and Z represent the laboratory system and (y, z) are the detector coordinates. The incident beam is parallel to X-axis. Each Laue spot has a specific position at the detector active area (z_i, y_i). ϕ is the angle between two diffracted spots and SDD denotes the distance between the sample and the detector. In some experiments, the sample could be moved in directions X, Y and Z with respect to the chosen reference position. The detector could be moved in three directions.

2.3.2 A typical EDLD pipeline

One of the most major advantages of the EDLD is that it has a systematic data-pipeline. It defines what, where, and how datasets are collected and driven. Employing an efficient pipeline, one can significantly boost the automatization of the data-wise processes (i.e. extracting, transforming, validating, and loading data) for further analysis. Figure 2.3 shows the basic layout of the conceptual EDLD process pipeline. The system design is threefold:

1. The hardware stage: it is the detector (i.e the pnCCD) operated in a frame-store module allowing for readout frequencies up to 1kHz in binning mode and 600Hz for the entire image [38]. It is served by a fast frame readout chip (CAMEX). The signals captured by the CAMEX are, then, converted into a digital form by the ADC unit and stored in the shared memory.

2. The data analysis stage: it is a problem solving technique which performs many consecutive steps towards the required application. It is a user-oriented tool to investigate and interpret the datasets collected by the hardware. It may be structured as a software package or a cascade of individual algorithms. The behavior and the design of such systems depends on the purpose of the experiment.

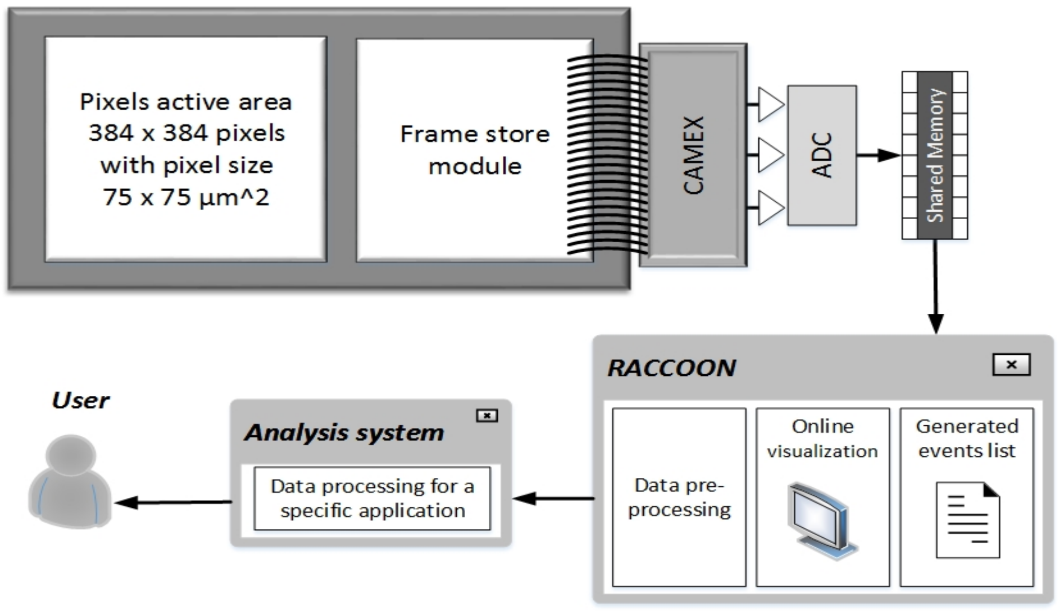


FIGURE 2.3: The layout of the data pipeline for the EDLD. The hardware layout is composed of a 2D energy dispersive detector (pnCCD) including the 384 x 384 active area, the frame store module and CAMEX readout chip. It is attached to the Analog to Digital Converter unit (ADC) and the shared memory. The software part has two components: (1) The Raccoon, which has a full accessibility to the shared memory and is considered as raw data delivery point (2) Analysis framework connected with Raccoon to execute data treatment and analysis.

3. The mediator: it is appointed to communicate between both hardware and analysis stages. It has a full access to the shared memory and functions as a data-truck delivering datasets to the analysis system. The currently used mediator is named *"Raccoon"* and is responsible for data preprocessing and basic online visualization [39]. It is supported by X-disk module to generate a list of the streaming data to be delivered to the analysis system. Since there is no unified analysis software, the current mediator is the final destination of the generated datasets, which are stored in disk-based databases. Further using or transferring of any dataset should be offline and manually performed.

2.3.3 EDLD data-handling system

The main aim of the EDLD data-handling system is to ensure that datasets are securely preserved and archived. Besides, it is responsible for integrating and interpreting the collected data in a manner that preserves the research project goals. After a sequence of detector-corresponding data-preprocessing steps (e.g. offset map, noise map and common mode noise) [32], the collected dataset is gone through different stages mentioned below.

2.3.3.1 Event-reconstruction

The event-reconstruction, often called event recombination, is the first station of the data, in which the recorded individual photons events are evaluated and categorized in three types (valid single-pixel event, valid multi-pixel event and invalid events)

depending on splitting direction and the spreading area within the pixel array of the detector [32]. For simplicity, when the system is operating in the single photons counting mode, only few photons are recorded simultaneously and not more than one photon is triggered by one pixel during each frame time interval. In this case, the number of the generated electrons in the charge cloud depends only on the energy of the absorbed photon. The charge cloud expands due to the drift and diffusion of electrons, passing through the silicon window towards the front end. The final size of the charge cloud, at the pixel plane, depends on the photon energy and the absorption depth within the depleted silicon substrate. The photon impact is localized at one pixel (single), two (doubles), three (triples) or four (quadruples) pixels, depending on the relative position and the size of the generated cloud [40]. Singles, left and right doubles, up and down doubles, four types of triples and four types of quadruples result in 13 possible patterns, shown in figure 2.4. The patterns which cannot be described by one of these 13 types are obviously not created by a single-photon impact. They are considered as invalid events and are named as *"fall-out"* events. These events may happen if two or more photon events are appearing close to each other. Figure 2.5 shows examples of such events recorded during an experiment. The dashed frames represent the borders of each event. Each event is a combination of two or more single-photon events shown in figure 2.4. Figures 2.5a and 2.5b display twofold events, while figures 2.5c and 2.5d show threefold combination. So far, there is no effective technique to resolve the *fall-out* events instead, they are denoted as *"unknown"* and their contribution is neglected by conventional analytical tools. Statistically, these events may represent up to 40% of the total triggered events, depending on the beam flux and the readout system frequency. This shows a demand to improve the event classification and to determine the actual number of photons detected during the experiment, which may limit the outcome of subsequent data treatment. Thus, many applications that demand extremely precise photons intensity calculations (i.e. structure factor analysis) are difficult to be performed due to involvement of multiple complex correction steps. For further extension of pnCCD applications, the issue of *"fall-out"* events has to be addressed.
In order to achieve a successful events-reconstruction process, frames data should undergo the following steps:

- Photon event pattern classification: the events are categorized in five types (single-pixel events, three classes of multi-pixel events and fall-out events).
- Events reconstruction: the multi-pixel events are reconstructed to individual photon hits by determining the center-of-mass coordinates of each event. A detailed study has been done in this direction [40].
- Fall-out events resolving: events that assigned as fall-out should be analyzed into a combination of two or more fundamental single photon events (i.e the thirteen types).
- Co-reconstruction: in this step, all resolved fall-out events are then reconstructed.

2.3.3.2 Data mining and selection

Data mining and selection represent a step, in which only a set of specific information is assigned for further execution, depending on the desired application, and the rest is neglected. In typical EDLD experiments, it is expected to see a Laue pattern

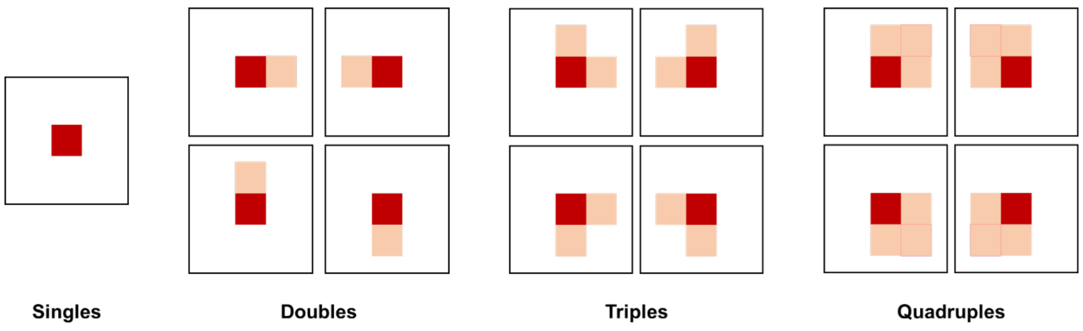


FIGURE 2.4: The thirteen possible event types originating from the interaction of single photon with the pnCCD detector.

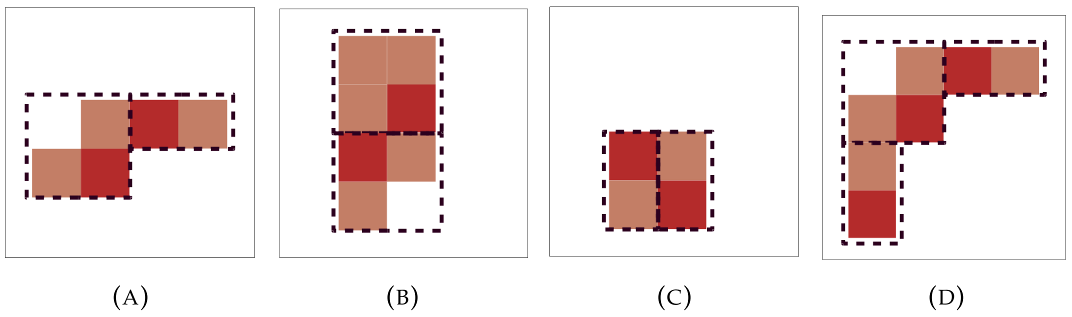


FIGURE 2.5: Different example of the *fall-out* events. (A) represents a double-triple combination, while (B) is a quadrapole-triple one. (C) shows a fall-out event of two double events. (D) is a threefold event consists of two double events and a triple event.

which contains several items that should be defined as background or foreground. As a key example, figure 2.6 shows a part of the integrated image from an EDLD experiment collected by using synchrotron radiation and a GaAs sample. It contains three different items. 1) The direct beam signal generated by the primary X-ray beam (i.e unscattered beam). The presence of such high-intensity signals may significantly affect the final results as it dominates the pattern, impeding the visualization of the Laue spots. Thus, the direct-beam should be segregated before any further calculations. 2) The background which is caused by many factors, such as air-scattered X-rays, intrusion of impurities within the setup and the electronics, and the detector dead pixels. These continuous signals should be eliminated to prevent signals-overlapping and the inconstancy in the final results. 3) Laue spots which are the areas of interest as they contain all details about the examined sample. They should precisely localized and analyzed. Beside their 2D position, relevant parameters have to be extracted from the recored dataset, namely energy spectra and intensity. Moreover, the shape of these spots may be an essential parameter for several applications, such as stress-strain analysis and grain deformation determination.

2.3.3.3 Data interpretation and visualization

Data visualization is the generation of visual representations of the recorded data. Many analytical graphics and statistical plots are created to give the user insights into the processed data. It may include charts, graphs, histograms, and 3D plots.

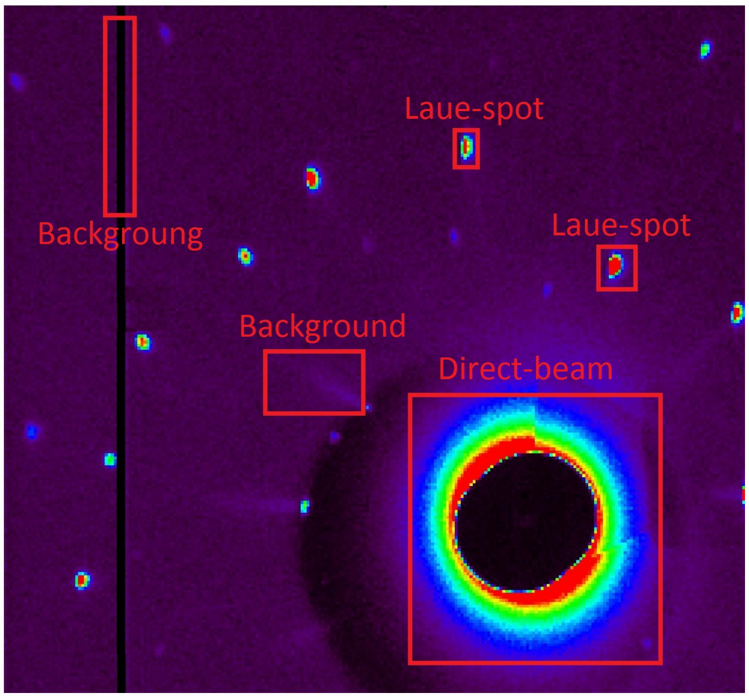


FIGURE 2.6: A part from a generate Laue patter from a GaAs sample.

Data visualization is particularly useful for experiment-steering, providing an effective indicator of the experimental procedure. A successful visualization tool for EDLD experiments is to be able to visualize data from different sources very fast and with confidence.
Data interpretation is a step, in which many physical equations and statistical relations are applied on the recorded dataset to convert the information into meaningful physical parameters, depending on the desired application. Creating an effective interpreting-tool usually requires a full understanding of both the scientific and the technical concerns, ensuring a digitalization and automation of the analysis procedure with the minimum user-interfere. The systematic data interpreting procedure of the EDLD facilitates the developing and deploying of different tools and platforms. A detailed overview of this procedure is shown in appendix C.

2.3.4 EDLD challenges

Datasets collected by an EDLD are profitable and can help the community to achieve the maximum interest in scientific research. However, achieving real-time data-handling is one of the most critical challenges because of the huge data volume collected during the experiments. Moreover, the complicated structure of the raw datasets make it difficult to implement new big data strategies. In order to deal with such challenges and pursue their analytics goals, EDLD-scientists have been trying to deploy or develop different tools and algorithms that perform individual tasks and require user-observation. However, it has been always a dream to consolidate all steps in one unified tool that can execute all data-handling tasks in a dynamic manner with high precision, low latency, and powerful performance. Technically, without such a coalesced platform, this field may fail to keep pace with the data growth.

Chapter 3

Big Data Fundamentals

"Big Data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it."- Dan Ariely

Last decade has witnessed a rapidly increase in data volume collected by different domains. The rise of these immense datasets introduced a new concept, namely; "*Being Data-Driven*"[41]. A data-driven organization, community or enterprise exploits and leverages data in order to efficiently develop new products or monitor the competitive surrounding. Therefore, the term "data science" has appeared in many contexts studying where information comes from, what it represents and how it can be turned into a valuable resource. This field makes use of such disciplines as mathematics, statistics and computer science, and incorporates a variety of techniques like machine learning, cluster analysis, data mining and visualization. Among all the buzzwords related to data science, *"Big Data"* is one of the most often heard [42]. General speaking, there is no consistent definition of this term. However, there have been many attempts to define Big Data based on its key features and incentives [41, 42]. This chapter presents a collective theoretical review, covering the bare essentials of Big Data frameworks, models, and scientific applications. It is organized in the following way: section 3.1 and 3.2 introduce the basics of the Big Data. Section 3.3 explains the data clouding technology, covering the cloud solution types and service models. Data warehousing and databases are presented in section 3.4, while section 3.5 is devoted to address the application of Big Data in crystallography community. This chapter is, then, concluded by the analysis of the developed system and an outlook for further works, shown in 3.6.

3.1 5V's theory

Before diving into the guts and glory of Big Data, it is necessary to briefly discuss its main characteristics, which are often described using five V's [43, 44]:

1. Volume: refers to the amount of data gathered by a domain. The term "big data" can be defined as a dataset that becomes so large that it cannot be processed using conventional methods. Therefore, more advanced techniques and algorithms are required not only to manage a huge data volume but also to make intelligent use of its size as efficiently as possible.

2. Velocity: refers to the speed at which Big Data are being generated, collected and analyzed. For time-sensitive domains (i.e. real-time responses), the analysis of data flow is a demand at an ever-increasing pace.

3. Variety: refers to the different types of data which come from diverse sources, including structured data (i.e. encrypted in one format) and unstructured (i.e having no predefined manner). In fact, unstructured data, such as images, videos and documents, represent 80% of the global data.

4. Veracity: refers to the quality and trustworthiness of the data. Many factors affect data accuracy, such as noise, software bugs, data lineage, and abnormalities. More specifically, it's not just the quality of the data itself but how trustworthy the data source, type, and processing.

5. Value: refers to the worth of the data being extracted. It is vital to understand the costs and benefits of collecting and analyzing the data to ensure ultimately that the data reaped can be monetized.

3.2 Data disciplinary

The challenge anyone faces while attempting to describe the process of data analysis is that, there is no regular basis for its explanation [45]. Introducing a compact framework involves characterization of the data analysis is an alternative to find the commonalities among different analytical approaches. From this perspective, this sub-chapter demonstrates a general process that can be applied in various cases, aiming to produce coherent results. Figure 3.1 displays the five main steps of data analysis framework.

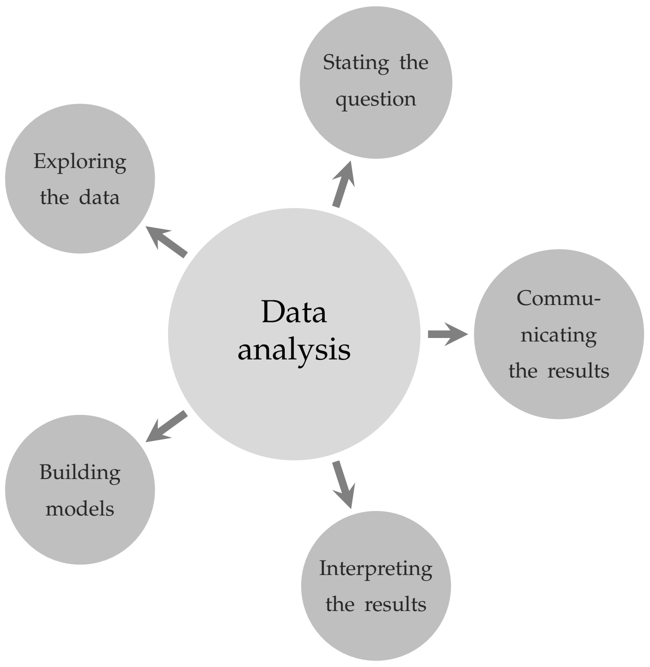


FIGURE 3.1: The diagram of data analysis framework

3.2.1 Stating and refining the question

Before undertaking any effective data analytics procedure, prior data-wise questions should be formulated in a way that is consistent with federal standards and organizational purpose. The main aim is to provide the data scientist with descriptive, inferential, and mechanistic summary of the dataset. This step helps the decision makers to set the scene for a well-organized approach to achieve the demanding output.

3.2.2 Employing Exploratory Data Analysis (EDA)

Exploring the data entails checking its structure and components. It also includes the distributions of variables and the relationships between two or more variables [46]. The goal of EDA can be summarized into the following key points [47]:

- Estimating assumptions based on which statistical conclusion will be reached.
- Suggesting hypotheses out of observations and their possible causes.
- Providing a logical reasoning for further data collection through surveys, investigations, or experiments..

3.2.3 Building formal statistical models

One of the main crucial purposes of formal modeling is to develop a precise specification of the question and to design an instrument aimed at answering that question based on the observations in research [48]. Moreover, modeling can be used to predict future observations. According to quantitative science, models fall into two groups, namely [49];

- *Deterministic models:* in which the relation between the input and output of the model is conclusively determined by the parameter values and the initial conditions.
- *Stochastic models:* in which the system contains one or more stochastic element. Generally, these models are not solved analytically as they possess some inherent randomness.

3.2.4 Visualization the results

Data visualization is a procedure enabling setting interaction between the user and the output of the processed data. It is an important part of data science which has been rapidly rising for last decades. Technically, visualization is the use of computer graphics to generate visual images which help to understand complex and massive data representations [50]. Big Data brings new challenges into this field because of the speed, size, and diversity of data that must be taken into account. Traditional architectures and software tools are not able to visualize massive data in a timely manner. Thus, innovative approaches should be adopted to address the challenges associated with visualizing Big Data.

3.2.5 Communicating and interpreting the results

Communication is the backbone of any successful data analytics procedure, as there is no point in doing any analysis if the process and the results are not being communicated to the audience. Data analysis tends to be extremely subjective. That is to say, the nature and goal of interpretation will vary from business to business depending on the type of data being analyzed and the audience receiving the output.

3.3 Data Clouding

The idea of data clouding was born in the 60th's of the 20th century due to a publication by D. Parkhill [51]. Data Clouding means providing a centralized computing facility as a public utility which is remotely accessible over networks [52]. It assumes that every software or hardware component represents a piece of the cloud. Recently, cloud computing has become a virtual part of any organization technology or business model. It offers an optimal solution regarding not only how to operate the infrastructure, but also how to save costs and authorize the third-party providers. This sub-chapter provides an overview on cloud solution types and service models, followed by a systematic example showing a medium-sized cloud.

3.3.1 Cloud solution types

3.3.1.1 Public Cloud

Public Cloud indicates a cloud type in which the services are being delivered online (i.e. via the Internet). The offered services may be free or subscription-based with costs depending on the amount of resources consumption (i.e. a pay-per-use basis) [53]. The cloud providers are in charge of developing, managing, and preserving the computing resources across the cloud network. The major benefits offered by the public cloud services are the following:

- Costs reduction
- High scalability and flexibility.
- Providing universal accessibility
- Automatic backup for data and applications

Despite the tremendous benefits offered, public cloud services suffer from security issues. Enterprises or organization cannot rely on the use of public clouds without taking some security considerations, namely;

- Security liability: the responsibility is split between the provider and the customer. The degree of responsibility depends on the type of cloud model.
- Data retention: data remnants are moved or deleted not to expose sensitive data to unauthorized sources.
- Multi-tenancy hazards: sharing nature of public clouds may lead to security risks such as unauthorized data access by other users, who utilize the same hardware.
- Compliance regulations: different regions have different data privacy regulations. It is important to consider the requirements of the region data can reside.

3.3.1.2 Private Cloud

Private Cloud refers to a cloud solution dedicated to be used by a single organization. The computing resources are separated and delivered by using a secure private network, and are not allowed to be shared with other tenancies. The provider takes the responsibility for creating private cloud environment, implementing, securing, and controlling the Cloud infrastructure [54]. The advantages of private cloud services are listed below:

- Flexibility to transform the infrastructure
- High scalability and efficiency
- No compromising on security and performance
- Dedicated and secure environments

However, this solution has several disadvantages, which can be summarized as follows:

- High total cost of ownership, compared to a public cloud
- The cloud stability is limited to on-premise computing resources
- Limited accessibility for mobile users

3.3.1.3 Hybrid Cloud

Hybrid Cloud is such a type of infrastructure that is a combination of the versatility provided by public cloud and the comfort level offered by private cloud. The resources are coordinated as an integrated infrastructure environment. Hybrid cloud computing is about aggregation of capabilities and services from cloud service providers with on-premises resources, leveraging the best-of-breed [55].The advantages of hybrid cloud services include:

- High reliability
- Flexible distribution of workloads across public and private environments
- Scalability of public environments without security risks

As far as the disadvantages of hybrid clouds are concerned, the main of them are enumerated below:

- Higher costs compared to a public cloud solution
- Extra compatibility and integration required due to the usage of two different solutions
- Additional complexity of infrastructure

3.3.2 Cloud Computing Service Models

3.3.2.1 Infrastructure as a Service (IaaS)

In this model, the cloud vendors provide the user with hardware and middle-ware resources [56]. According to the National Institute of Standards and Technology (NIST), IaaS is a model provides physical assets to consumers to support data processing, storage, networks, and other fundamental computing issues [57]. This form enables the user to implement arbitrary software resources (i.e. operating system and applications). Moreover, it gives the user control over the storage and the deployed applications. On the other side, the consumers cannot manage the underlying cloud infrastructure.

3.3.2.2 Platform as a Service (PaaS)

PaaS model provides a development environment to deploy onto the cloud infrastructure acquired and applications created by the consumers [57].The cloud provider has to support programming languages used as well as, libraries, tools, or services. The user has the right to manage the created applications and configuration settings of the hosting environment. However, it is forbidden for the consumer to control the underlying infrastructure, servers, operating system, or storage.

3.3.2.3 Software as a Service (SaaS)

SaaS model is located on the top layer of the abstraction layers [56]. Its capabilities make the consumers independent of their own resources. It enables the users to access databases and softwares using applications hosted by a vendor [57]. In fact, SaaS is considered as "On-Demand" software, in which the customer has access to a single copy of the provider-created application. It allows the provider to integrate softwares using Application Programming Interfaces (APIs). In addition, new functionalities and features of the application might be developed in much faster and more feasible way, as the source code is shared for all customers.

3.4 Data warehousing and In-Memory databases

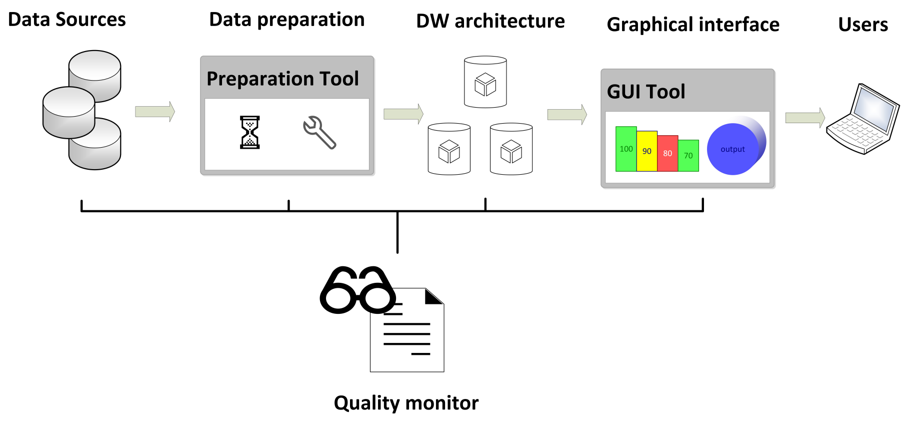


FIGURE 3.2: The environment of a data warehouse.

Data warehousing is a method that aggregates data from operational systems and external data sources into one central repository, so-called *"Data Warehouse (DW)"*. A DW is a relational database that contains properly integrated, cleansed, and recoiled data from disparate sources, aiming to make the data available for analysis, comparison, and evaluation. It should be subject-oriented, non-volatile, historical, and summarized. The generic environment for a valid DW contains four components, namely: data source, data preparation and integration tool, data warehouse architecture, and user-interaction tools. Figure 3.2 shows the pipeline of the DW-environment components. The environment is usually supplemented by an extra component, so-called *"Quality Monitor"*. Its main aim is to ensure that the DW meets its requirements.
Typically, the physical storage space for a DW deploys many hard disks as hosting

databases (e.g. disk-based). However, such storages suffer from considerable drawbacks regarding data accessibility and loading time, as the lay at the bottom of the memory hierarchy. Figure 3.3 shows the memory hierarchy for different memory types. The CPU-wise memory types give the highest performance and lowest latency but they are costly and support very small space occupancy. In comparison, the physical hard disks offer a huge storage volume and low costs but lower performance and higher latency. The main memory is an intermediate element, which balances all the memory components.

3.4.1 In-Memory Databases (IMDB)

In-Memory Database (IMDB) is an alternative solution to avoid latency which is typical of disk-based databases. The usage of in-memory computing and data storage options might be widely applicable for designing the next generation of data warehouse systems. An IMDB defines storage, where data is primarily stored in the main memory of the device (Random Access Memory (RAM)). Compared to other databases, IMDB gives considerable advantages, such as performing computation with a lager amount of data in less time as well as improved access to the data.

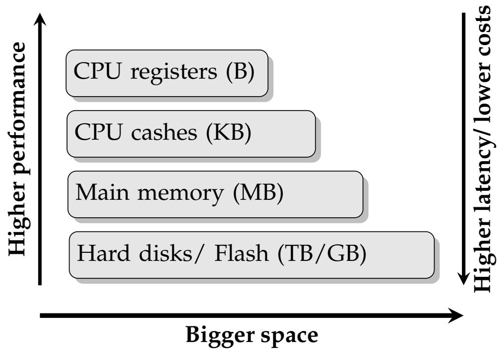


FIGURE 3.3: The memory hierarchy

On the other side, the main memory is volatile, since there is a risk of losing all the data if the power is removed. It thus impairs the non-volatility of the DW. Moreover, it might not be possible to use the same IMBD for different systems and architectures, which causes results in the need to make some changes in the program code, resulting in additional costs.
A proper solution is to use the conventional database management systems, such as Relational Database Management System (RDBMS). It relies on both the physical hard drive and the RAM as storage. When a query is requested the data from physical hard drive are sent to main memory (RAM) for further processing [58]. In contrast with IMDB, the cost of memory is drastically low.

3.4.2 Cloud-based Data Warehousing (CDW)

Cloud-based warehouses allow the users to modernize their processes as quickly as a new technology is developed. Moreover, they simplify data access for the entire organization. Communities and enterprises have been increasingly moved towards cloud-based data warehouses, leaving traditional systems. The main differences between cloud-based data warehouses and traditional on-site warehouses are the following:

- No need to purchase physical hardware or or upfront licensing.
- Faster performance due to the usage of Massively Parallel Processing (MPP).
- Easy to be set and scaled

3.5 Application of Big Data for the X-ray crystallography community

The X-ray crystallography community has recently been affected by a significant increase in data volume caused by the use of advanced detector technologies. The fact that forced the decision makers to implement Big Data and Big Data analytics, aiming to achieve a suitable environment for scientists at experimental and post-experimental phases. This sub-chapter introduces a new approach to use warehousing and cloud computing to manage datasets collected by 2D energy-dispersive detectors, for an example. Moreover, it suggests that, deploying a Software as a Service (SaaS) cloud model, a public cloud data center, and cloud-based warehousing architecture, it is possible to dramatically reduce both hardware and processing costs.

3.5.1 Data history and nature

Since 2007 the MPI-halbleiterlabor and the University of Siegen have initiated a collaboration to develop a new non-destructive testing (NDT) method, which can be applied in different fields (e.g. Crystallography, Material science, Nanowires technology, and Mechanical engineering). Meanwhile, an enormous amount of precious data has been collected from various materials, radiation facilities, and geometrical setups. The processed data have demonstrated the reliability of this technique as a promising procedure for material microstructure characterization. However, the traditional approach of data encapsulating and storage is one of the biggest challenges to keep pace with *"the era of real-time"*. For instance, a typical one-shot raw image data ($\approx 50 \times 10^3$ frame) consumes around 20 Gbytes of the storage volume. This amount is explained by the fact that, each frame is statically saved as a matrix with a size equal to the number of detector pixels. The space required for one frame is number of pixels multiplied by the size of unsigned integer (2-4 bytes), regardless of the pixel status (i.e. is lighted or not). Conceptually, each frame is stored as a large sparse matrix (i.e. in which most of the elements are zero) with extra unnecessary details, such as frame time stamp, detector ID, and related files' names. Moreover, all data elements are encoded into a complicated format, requiring a specific decoding algorithm. Seemingly, dealing with such raw datasets (i.e. streaming, decoding and refinement) is intractable and time-consuming (e.g. requires 8 hrs/image) [10].

3.5.2 From Sparse Matrix to Coordinate format (COO)

Very fast readout systems and high X-rays energy, used in the crystallography community cause the reduction of the pixel detector occupancy per recorded frame. The number of events recorded per frame for a typical EDLD experiment is around $\approx 300 - 1000$ event, which represents less than 1% pixel occupancy (with detector area 384×384 pixel). Technically, sparse matrix is easy to be handled within a software and efficiently parallelized. However, storing large matrices with high sparsity (e.g. greater than 50%) is infeasible and storage consuming. Instead, the concept of storage Coordinate formate [59] has been introduced to be the standard formate for any dataset saved within the cloud. The COO is a particularly simple storage scheme, where only the row indices, column indices, and values of the nonzero matrix elements are considered [60]. Generally, COO is a sparse matrix representation technique, by which both row and column indices are explicitly stored. It shows two main limitations, namely; 1) No static size since, the required storage is proportional to the number of nonzero elements. 2) Requires special algorithms to conveniently process this format. Figure 3.4 illustrates the traditional representation of an example 7×7 matrix and its alternative COO representation. In traditional implementation, as displayed in 3.4a, all elements (e.g. nonzero or zero) are represented. On the other hand, as shown in 3.4b, only the nonzero elements are represented in COO formate, ensuring that entries with the same row index are stored contiguously [10]. In order to practically achieve that, the streaming module, which is responsible of acquiring and writing the data, has been updated to generate COO formate as the standard data formate. Moreover, all analysis tools have been equipped with the required algorithms to read and treat this formate. More details about the implementation and the testing are given in chapter 4.

$$\begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 8 \\ 1 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(A)

$$Value = \begin{bmatrix} 4 & 8 & 1 & 6 & 3 & 7 & 3 & 9 & 2 \end{bmatrix}$$
$$Col = \begin{bmatrix} 0 & 0 & 0 & 4 & 2 & 6 & 0 & 4 & 2 \end{bmatrix}$$
$$Row = \begin{bmatrix} 0 & 6 & 1 & 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

(B)

FIGURE 3.4: The traditional sparse matrix representation for a snippet 7×7 matrix vs. its COO representation. (A) is the sparse matrix representation, while (B) is for Coordinate representation.

3.5.3 The cloud

The University of Siegen is equipped with a cloud infrastructure , so-called *"ScieBo"*. This cloud offers a free service for all members of the university. *"ScieBo"*, which name is formed due to blending word is created from of parts of two words, i.e. "science" and "box", is a data storage for all the universities in North Rhine-Westphalia, enabling users to save, synchronize and share data [61]. The storage capacity is up to 500 GB for each employee with additional storage volume for research data. The stored data are protected by the Federal Data Protection Act (BDSG).

3.5.4 The main framework

The main framework describes a particular SaaF environment which gives the user a full leverage over the whole data-processing cycle. It includes a set of tools, APIs and algorithms which controls all analytical procedures required for a successful experiment, such as data streaming, standardization, saving, restoring, and analysis [10]. The workload is distributed between three components, namely: Data Manager, On-demand warehousing, and analysis tools. Figure 3.5 displays the pipeline of the main framework. The Data Manger is surrounded by blue box while the red and the green boxes envelope the On-demand warehousing and analysis tool, respectively.

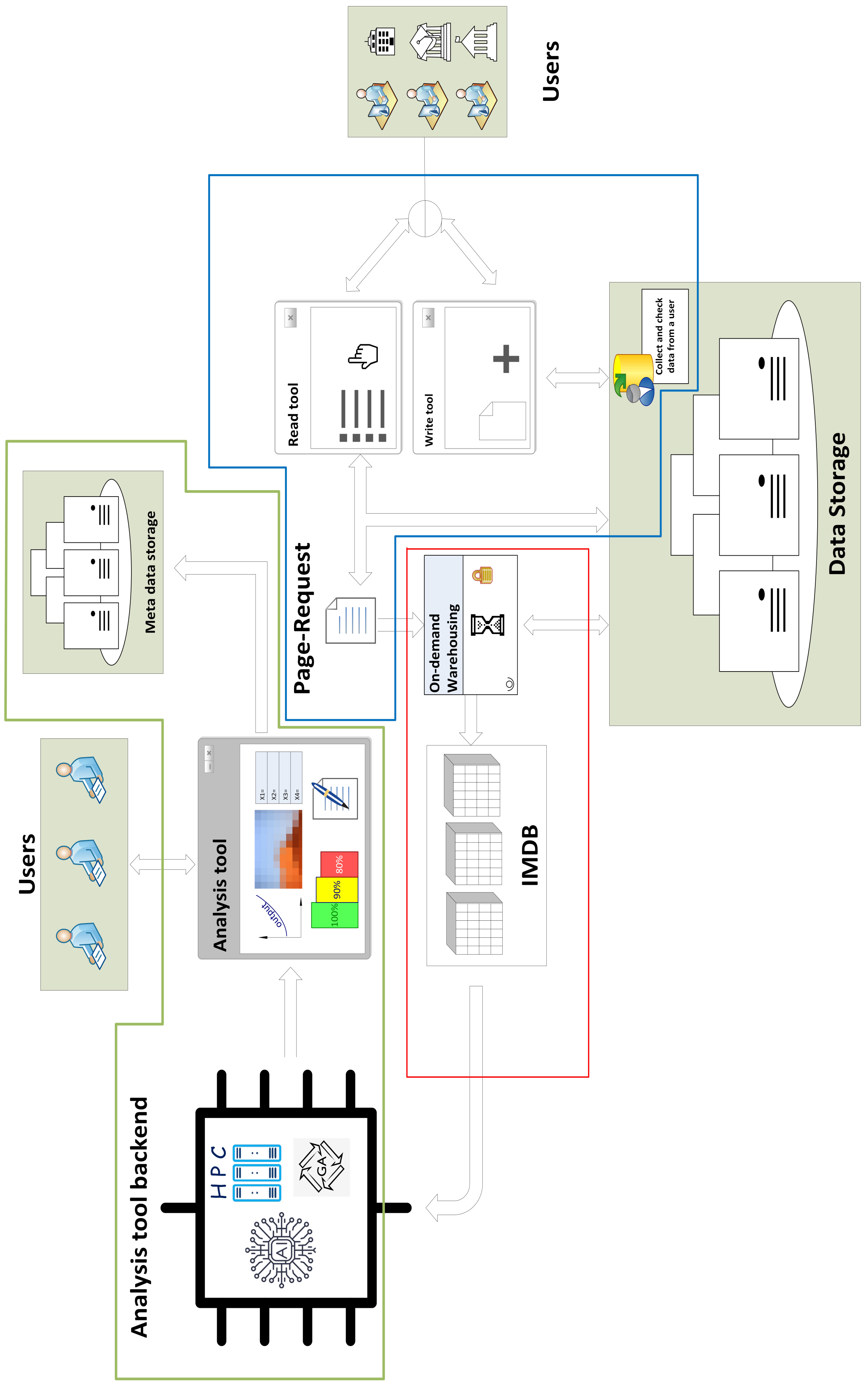


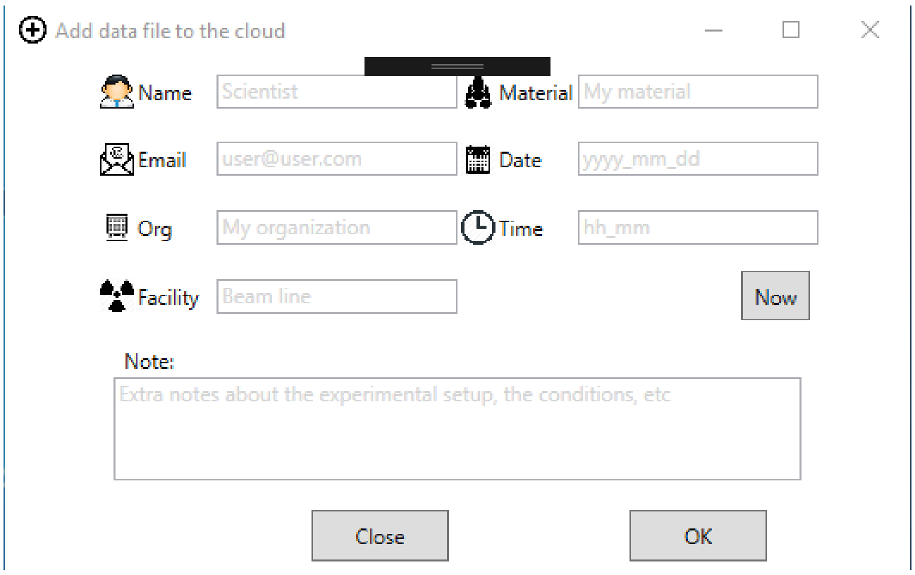
FIGURE 3.5: The main framework.

3.5.4.1 Data Manager

Data Manager is a tool designed for adding, operating, accessing, and monitoring data in the cloud. It ensures that data operating is optimal and the system is properly interacting with users. It has two main tasks, which are to add new datasets to the cloud and to release datasets for analysis, depending on the user call. Furthermore, it provides the users with the connection channels to data storage. In order to simplify this aspect, a decision has been taken to give each object (i.e. data file) an arbitrary Digital Object Identifier (DOI). DOIs are alphanumeric strings having a unique and persistent reference to a specific object. They are bound with the object's metadata, indicating the path to the directory where the object can be found in the cloud. Aiming to standardize DOIs, a special tool has been implemented to generate a DOI required to store a file within the cloud. Figure 3.6 shows the GUI for desktop application of the implemented tool to generate the standard DOI. This tool is the only writing tool which has the authority to add any new data file to the cloud. The user has to fill the required description items about the experiment (i.e. name, date, material used, radiation facility, organization, etc). After writing the required items, the tool will automatically generate a DOI for this file and store it in the cloud. Moreover, a metadata file, so-called *"README"*, will be generated. This file contains all the details provided by the user, giving an overview on the experiment to other users.
Data Manager is also equipped with a releasing tool, which is responsible for viewing, accessing, and sending stored data files to be analyzed. Figure 3.7 displays the GUI for a desktop application of the releasing tool with snippet data files, in which each row represents a data file with columned description items. The tool provides a search bar, enabling the user to find a specific file within the cloud. Thus the user is able to select one or more files to send them for further processing. The DOIs of the selected files will be sent as a queue, so-called *"Page-Request"*, to the On-demand warehousing tool. This request is the final output of the Data Manager.
The tool is based on the Model–View–ViewModel (MVVM) architecture pattern, originated from Microsoft, which is specialized in the Presentation Model design pattern [62, 63]. It is a managed Windows Presentation Foundation (WPF) subsystem, the front-end of which is written in XAML, while its back-end is written in C#.The system has been supplemented by a Message Bus System (MBS) to synchronize GUI threads and to route the traffic of the registered subroutines. The web application version is developed in the .NET Cross-Platform [64] and is still in the testing phase.
Technically, Data Manager is a managed code that compiles to the intermediate language and is executed using Common Language Runtime (CLR). On the other side, On-demand warehousing is an unmanaged code that compiles directly to machine code. Therefore, binding codes, based on the C++/Common Language Infrastructure (CLI), are implemented to enable communication between both tools [65]. These codes are responsible for data marshalling between the concepts.

3.5.4.2 On-demand warehousing

On-demand warehousing is a memory allocation and -deallocation tool, which is responsible for reserving and freeing memory portions for the required data. It implicitly transfers data from the cloud warehouse to the In-Memory database upon the user request and frees this database after completing the analysis. The *"Page-Request"* queue sent by Data Manager is the input of the On-demand tool. The

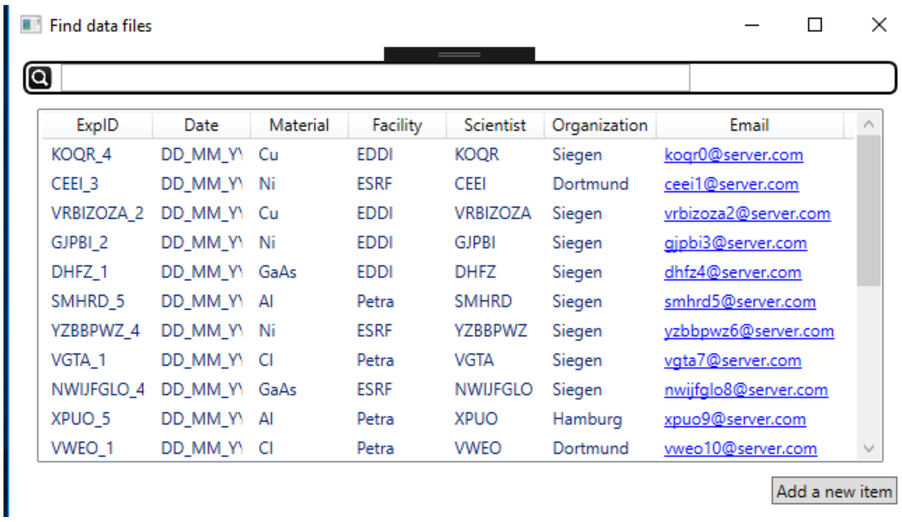

Add data file to the cloud

Name	Scientist	Material	My material
Email	user@user.com	Date	yyyy_mm_dd
Org	My organization	Time	hh_mm
Facility	Beam line		Now

Note:

Extra notes about the experimental setup, the conditions, etc

FIGURE 3.6: The GUI of the DOI generator tool.


Find data files

ExpID	Date	Material	Facility	Scientist	Organization	Email
KOQR_4	DD_MM_Y\	Cu	EDDI	KOQR	Siegen	kogr0@server.com
CEEI_3	DD_MM_Y\	Ni	ESRF	CEEI	Dortmund	ceei1@server.com
VRBIZOZA_2	DD_MM_Y\	Cu	EDDI	VRBIZOZA	Siegen	vrbizoza2@server.com
GJPBI_2	DD_MM_Y\	Ni	EDDI	GJPBI	Siegen	gjpbi3@server.com
DHFZ_1	DD_MM_Y\	GaAs	EDDI	DHFZ	Siegen	dhfz4@server.com
SMHRD_5	DD_MM_Y\	Al	Petra	SMHRD	Siegen	smhrd5@server.com
YZBBPWZ_4	DD_MM_Y\	Ni	ESRF	YZBBPWZ	Siegen	yzbbpwz6@server.com
VGTA_1	DD_MM_Y\	Cl	Petra	VGTA	Siegen	vgta7@server.com
NWIJFGLO_4	DD_MM_Y\	GaAs	ESRF	NWIJFGLO	Siegen	nwijfglo8@server.com
XPUO_5	DD_MM_Y\	Al	Petra	XPUO	Hamburg	xpuo9@server.com
VWEO_1	DD_MM_Y\	Cl	Petra	VWEO	Dortmund	vweo10@server.com

FIGURE 3.7: The GUI of the releasing tool with snippet data files.

selected files are streamed and their data are allocated in the main physical memory. Each data file is supplied with a separated cube In-Memory database, which is considered to be the bulk data structures [66, 67]. After the allocation process is succeeded, the analysis tool kernel gets launched and the On-demand tool is sent to the *"on-hold"* mode. Pointers of the allocated data structures are sent to the analysis tool kernel for being buffered and analyzed. The deallocation process is performed upon receiving the *"DONE-MSG"* massage, which declares that the analysis kernel has successfully completed its task and is deleted.
The main aim of the On-demand tool implementation is to reduce the memory costs. The collaborative connection between the warehousing and analysis tools shows a sophisticated technique to control the memory consumption. Moreover, a combination of both the physical hard drive and the RAM can reduce the infrastructure hardware costs.

In order to boost the liability and robustness of the On-demand tool, several precautions have been considered:

- **Protection:** This tool is fully protected from the user's intrusion. Specifically, the user has no access to any component (i.e. libraries, functions, pointers, etc) within this tool. Moreover, all processes and procedures performed in a hidden manner. This prevents the system from any error or bug accidentally caused by the user.

- **Exceptions handling:** An exception is an error or unexpected event, which happens during the execution time. Such an event can interrupt the flow of the program. In case of the memory allocation and -deallocation tool, the most expected exception is an error called *"Bad Allocation"* (or bad_alloc). As a preventive measure, "**try** – **catch**" block has been implemented. All allocation processes are performed within the "**try**" block, which catches any exception. The "**catch**" block then represents the following procedure with a specific exception (e.g. bad_alloc in our case). Figure 3.8 display a snippet of the implemented exception handling method.

```
#include <new>                    // including std::bad_alloc

void memory_allocation (string file_path){
        try{
                \\ the memory allocation process
                }
        catch(std::bad_alloc &my_exception){
                \\ perform something to handle the exception
                }
}
```

FIGURE 3.8: A snippet code of the bad allocation exception handling

3.5.4.3 Analysis tool

Analysis tool is the final destination of the data pipeline. It is responsible for data processing, visualization, and interpretation. It makes use of several cunning and pioneer techniques, such as High Performance Computing (HPC) [68], Artificial Intelligence (AI) [69], and Genetic Algorithms (GA) [70]. The main aim of any analysis tool is to establish the communication channels between the user and the data processed. Moreover, all its components should harmonically and consistently incorporate to generate a meaningful output, fulfilling the user demands. The architecture and the logic of an analysis tool varies from system to another, depending on the demands, policies, and altitudes of the user and/or the organization. Usually, the analysis tool is connected to a sub storage, where metadata files are stored. This gives the user an access to the previously analyzed datasets, preventing the data analysis duplication. The following chapters will demonstrate how these techniques might be implemented to enhance the performance of the analysis procedure.

3.6 System analysis and conclusion

This part demonstrates the system analysis for two different cloud solutions, namely: the off-shore (public) and the on-premises (private) clouds. Table 3.1 shows the comparison of several parameters related to both solutions. The costs have been calculated by using AWS Total Cost of Ownership (TCO) calculator, provided by Amazon [71] for infrastructure configurations given in table 3.2. Figure 3.9 and table 3.3 display a detailed overview about the costs distribution for both cases per year. It is clearly seen that public clouds offer a low-price solution, compared with the private cloud, with factor up to 70%. Moreover, public solution shows variety of lucrative features, such as remote access, unified sharable resources, flexible tasks schedules, and low administrative workload. On the other side, on-premises systems are locally access and have limited applications-compatibility. Storage-wise, they prove more powerful than the public clouds. However, public systems demonstrate fast storage size growing.
It is essential to mention that, Amazon cloud has been chosen as an example for the public cloud system. The reason behind this decision is fame and wide range of this cloud. In addition, it show several similarities with the used cloud. Practically, cloud market is getting stronger and denser, increasing the competition between the cloud providers. Therefore, vendors are forced to aggressively reduce service costs. In such a competitive market, long-term customer leads to high market volatility. In such a volatile market, long-term customers are highly appreciated by the cloud vendors. Taking this advantage, big communities (i.e. the scientific community) are enabled to provide sustainable growth for their business with low cloud ownership costs.

TABLE 3.1: Comparison between cloud-based and traditional solution

	Public (AWS)	On-premises
Total costs	$\approx$ 94k €	$\approx$ 233k €
Storage size	Small	Large
Size growing	Fast	Slow
Accessibility	Remotely	Locally
Access model	Any device	Corporate Desktops
Applications	Limited	Unlimited
Servers	Integrated	Co-located
Resources	Unified	Partitioned
Tasks scheduling	Flexible	Planned
Administration	Reduced function	Overhead
Infrastructure	Can be subscripted	Self-establishing

In this chapter, the Big Data fundamentals were investigated. The potential of data clouding and in-memory warehousing for applications in the crystallography community could be demonstrated by the example of EDLD experiments. In the course of this work, various tools and algorithms were developed in order to obtain the convenient environment for scientists. It could be shown that utilizing such innovative techniques boosts the performance of the whole organization. In the case of big communities, exploiting SaaF model and a public cloud serves offers an optimal solution with low costs and high performance.

TABLE 3.2: Cloud infrastructure configurations

Configurations	Details
Server type	DB
# of processors/server	1
# of cores/processor	4
# of servers	10
Memory	16 (GB)
# instances	10
Storage type	SAN
Raw storage capacity	1 (TB)

TABLE 3.3: Costs details

	On-premises	Public (AWS)
Server	≈ 117k €	≈ 49k€
Storage	≈ 48k€	≈ 975€
Network	≈ 57k€	≈ 33k€
IT-Labor	≈ 10k€	≈ 11k€
Total	≈ 233k€	≈ 94k€

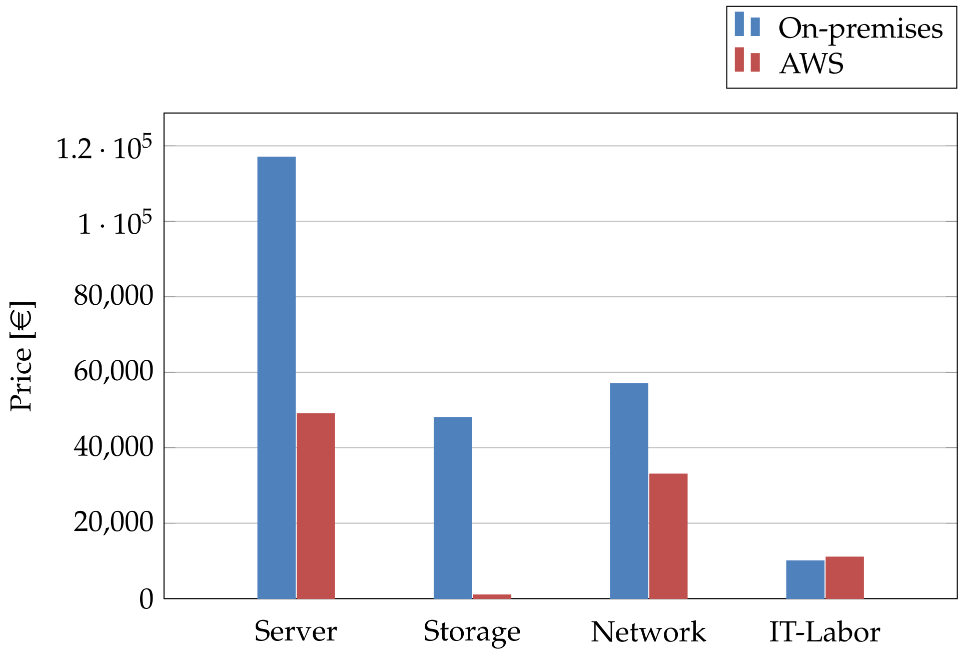


FIGURE 3.9: Comparison between the costs of on-premises and AWS cloud systems. The blue bars represent the on-premises solution, while the red ones indicate the AWS.

Chapter 4

High performance computing and parallel programming

"Redesigning your application to run multithreaded on a multicore machine is a little like learning to swim by jumping into the deep end."- Herb Sutter

4.1 Introduction

High-Performance Computing (HPC) is the phenomena that describe computing environments used to address problems that require sophisticated computational and significant processing time. Surprisingly, the original pertaining of computing was, in a way, high-performance computing. This term has started at the beginning of the 20*th* century with a slightly shallow definition: Anything can perform computation faster than a human being [72]. In 1920, this HPC has attained a great uprising to include a new term, namely; *"Supercomputing"* [73]. After a few decades, it was the first commercial implementation of the *"Control Data Corporation (CDC)"* as supercomputer mainframe [74]. Since that time, using clusters of independent processors connected in parallel has become a widespread trend. For many organizations and enterprises, HPC is an intrinsic side of any successful business. It is involved in many complex problems, such as designing products, structure or flow simulation, climate or geophysics simulation, and optimizing manufacturing. The scientific community, as well, considers the HPC as a main element of the equation. Therefore, many scientific facilities (i.e Large Hadron Collider (LHC) at CERN, ESRF) have established their own HPC section for better data handling and numerical modeling.
Generally, There is no clear definition of HPC term. A helpful way for better understanding of the HPC is to think about its anatomy. It can be structured into two components, namely: hardware and software. An effective HPC system is to have both levels homogeneously and effectively interact with one another to shape the demanded performance. In such manner, it is suggested that HPC might be integrated from three concepts:

1. Think smarter to melt the edges of the hardware by a genius software. The first level of HPC starts with the design of the execution algorithms. Rather than focusing on Hardware-resources, the leverage is built around leading-edge and providing concrete optimization software. That can help to adapt feasible environments exploiting the maximum hardware-resources, saving execution time, and reducing operating costs.

2. Work harder by involving more computing power. Almost all available processing units have multi or many-cores connected in parallel, where a core is

a processor that carries out instructions sequentially. Such architectures are suitable for many computing problems that can be partitioned into independent portions. In such manner, each processing core processes a portion of the problem in parallel, and then combining the final processing results for each portion. This type of computing is often referred to as *"parallel computing"* or *"data-parallel applications"* [75].

3. Get extra help by breaking computers into clusters. In its simplest structure, a cluster of computers incorporates computational powers of the processing cores to provide an enormous combined computational power. This paradigm is the base of the *"supercomputing"* [73].

This chapter demonstrates how the implementation of high performance and generic algorithms can help to significantly reduce the execution time and costs. Section 4.2 represents a comparison between the two hardware components that are used in this work (i.e. the Central Processing Unit (CPU) and the Graphics Processing Unit (GPU)). The sequential and parallel programming concepts are explained in section 4.3, while section 4.4 gives some suggested keys to design a successful algorithm. Finally, a practical model of HPC implementation in EDLD experiments is discussed in section 4.5 followed by the conclusion in section 4.6.

4.2 Central Processing Unit (CPU) Vs Graphics Processing Unit (GPU)

The CPU and GPU are distinct hardware components. They have different purposes but equal importance. Both the CPU and the GPU are embedded in silicon-based microprocessors systems. They work together to provide the domain with the demanded computing power.

4.2.1 CPUs

A CPU is often called the brain of any ingrained system. It carries out the instructions of a program by performing control, logical, and input/output operations. A CPU is consisted of millions of transistors and comprises three main components: 1) The Arithmetic Logic Unit (ALU) which is responsible of storing and performing the information. 2) The Control Unit (CU) is the scheduler that organizes and executes the instruction queuing. 3) Cache memory, often called the CPU memory, is high-speed Static Random Access Memory (SRAM) that is faster accessed by the microprocessors comparing with the regular Random Access Memory (RAM). The CPU communicates with the other hardware components through the Dynamic Random Access Memory (DRAM) that stores each bit of data. Figure 4.1a displays a scheme of the assembly of a single-core CPU.
Architecturally, the CPU has a few cores with lots of cache memory and it can execute a handful of operations at once [76]. The first CPU was developed by Intel in the 1970s [77]. Then for a few years most of the CPUs were designed with a single core. Owing to fast improving in chip design and manufacturing, the current generations of the CPUs, however, have between 2 and 28 powerful cores [78]. They are optimized to minimize the latency of a single thread. However, their memory consumption is high, unlike GPU. The CPU provides more effective results in case of processing serial instructions.

4.2.2 GPUs

A GPU is the brawn of the computing system. It is effective in performing much more work per unit of energy as it emphasis on high throughput, comparing with the CPUs. The first GPUs were optimized to offload and accelerate the graphical computations (i.e. the creation and rendering of images, video, and animations) [79]. Nowadays, GPUs are the key to different applications, such as automotive industry, health-care and life sciences, and artificial intelligence. [80]
GPUs, in contrast to CPUs, contain more ALU and smaller cache, with much higher bandwidth. A bunch of ALUs with a portion of the cache memory, is often called "*parallel data cache*" or "*L1 cache*", are coalesced together to compose a Streaming Multiprocessors (SM) [81]. A series of SMs, then, share a lower hierarchy cache memory, namely; "*L2 cache*". Figure 4.1b shows a basic diagram of a GPU. In the same manner as the CPU, a GPU incorporates with other system components by sharing RAM.

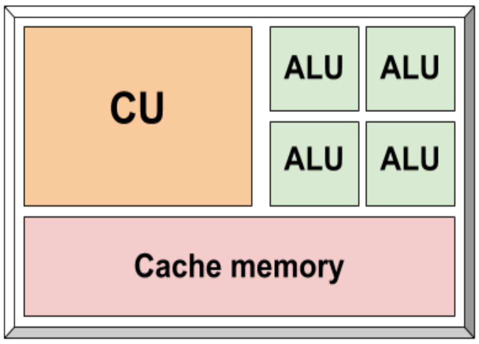
4.2.3 How do they Work Together?

As the CPU is the brain of the system, any assigned work to GPU is done through the CPU. Usually, the CPU assigns tasks to GPU and waits for GPU to finish them. GPUs and CPUs work independently as both generally execute different data and perform in unlike manner. Technically, CPUs do all the complicated logic processing, as long as the tasks only involve basic calculations. GPUs follow the principle of Single Instruction Multiple Data (SIMD) allowing for fast execution of more complicated calculations. Actually, GPUs are not alteration of CPUs but they are considered as accelerators for a specific task. A GPU-based application offloads compute-intensive tasks to the GPU, while the rest of the tasks are executed by the CPU. Usage of the bandwidth of both CPUs and GPUs may be monitored in the following examples:

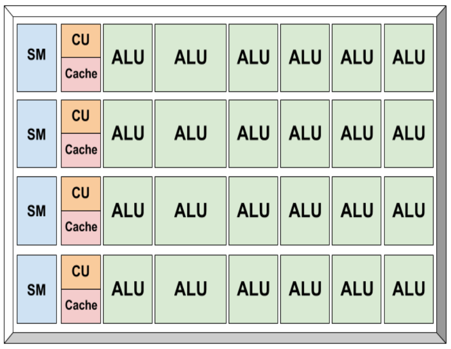
- Big Data: GPUs are the pioneering choice to power applications designed to perform tasks on Big Data. They are able to accelerate the amount of data a CPU can process in a given amount of time. because of their huge throughput and large number of computational cores.
- Artificial intelligence: Systems that are based on artificial intelligence require complex mathematical calculations which can be offloaded by the GPU. This frees up time and resources for the CPU to perform other tasks.
- 3D Visualization: Many 3D modeling and visualization applications and libraries, such as Computer Aided Design (CAD) and Open Graphics Library (OpenGL) rely on GPUs to draw, viewport, render, rotate or move those models in real time.

4.2.4 The bottleneck

A bottleneck is a point of congestion in the system. It occurs when too many workloads are assigned than can be performed and returned in a given time frame. Significant divergences in throughput between the CPU and the GPU may cause critical bottlenecks. Variances in the hardware specifications are the most likely technical issue. It may be a consequence of pairing a high-performance processor with a lackluster graphics card or vise Versa. Bottlenecks may also arise when the execution program fails to achieve a balanced distribution of workload between both components.



(A)



(B)

FIGURE 4.1: A scheme of the main hardware component assembly. (A) is the CPU while (B) is the GPU.

4.3 Sequential Vs Parallel programming

A sequential software is said to be *"one-thing-at-a-time"* [82]. It is like hiring one worker who carries out a step-by-step working plan, finishing one step before starting the next. Technically, a problem, to be solved by a sequential algorithm, should be broken into a discrete queue of instructions. The instructions are then executed sequentially one after another on a single processor. Figure 4.2 visualizes a scheme of the sequential behavior where a series of instructions is being performed stepwise. Only one instruction may execute at any moment in time.

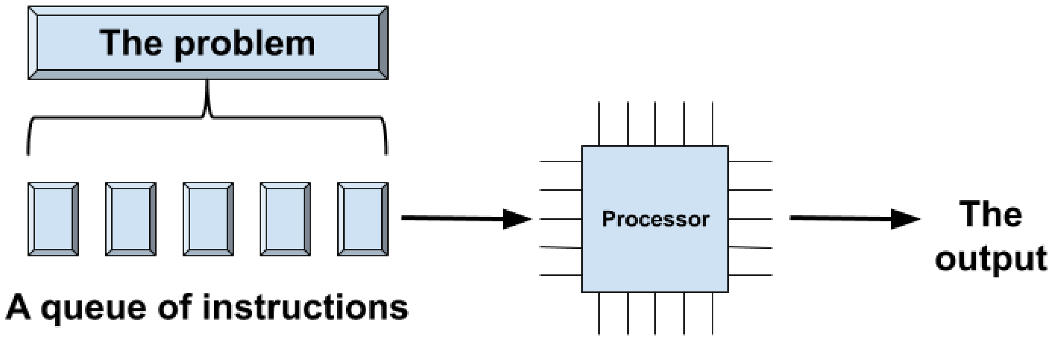


FIGURE 4.2: A scheme of the sequential programming behavior.

On the other hand, parallel programming violates the concept of *"one-thing-at-a-time"*. Hiring several or many workers to finish the assigned tasks introduces a new assumption: *"more-than-one-thing-at-once"* [82]. Whereby, all workers work simultaneously and each one carries out an independent task. In the simplest sense, a problem, to be solved by the parallel programming, is broken into discrete parts that can be performed concurrently. Each part is further broken down to a queue of instructions executed simultaneously on different processors. Figure 4.3 visualizes a scheme of the parallel behavior.

4.4 Keys to success

Building a successful software in HPC is a difficult task to be measure. Ultimately it depends on the goals, resources, and budget of the domain. This section paves the pathway to the strategies that can take complexity out of the analysis process.

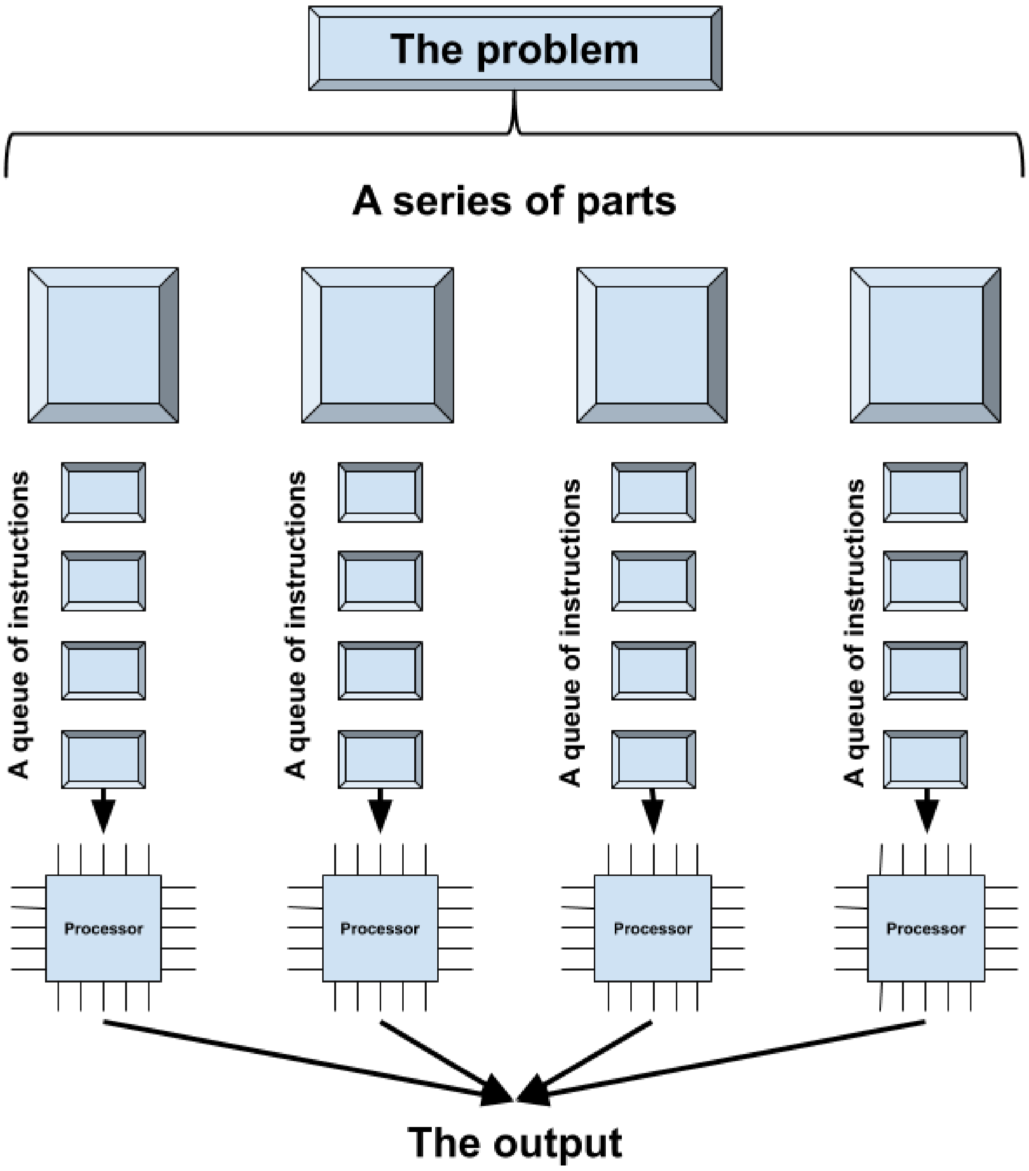


FIGURE 4.3: A scheme of the parallel programming behavior.

It suggests essential guidelines that make it manageable, successfully getting the results necessary to advance the application.

4.4.1 Build the scene

Prior to making any investment in software, some general considerations should be initialized: 1) Mapping out the aspects of the software and clear definition of every step taken to move product out the door. 2) Budget boundaries planning, including the ongoing support and maintenance costs. 3) Technical limitations addressing, in which the headway of the product is assigned based on the technical experience of the team. 4) Time constraints realizing, including the deadlines and the scheduled releasing time.

4.4.2 Costs of complexity minimizing

Software complexity has a huge impact on software acquisition costs. According to the software complexity costs study [83], it is suggested that software maintenance represents about 70% of the total acquisition costs. Maintenance cost is proportionally related to the software complexity. Increasing software complexity can significantly increase the maintenance costs by $\approx$ 25%. Thus, the ability to measure and control software complexity is of paramount importance. Generally speaking, complexity always has costs which can be a trade off with the value delivered by it. In order to minimize such costs, the following characteristics should be considered: 1) Learnability is the degree of ease that the user can understand and interface with the software. A learnable product should offer short or steep learning curves. In other words, it helps the user to learn how to use the software in less time and without having been previously trained. 2) Usability is defined according to (ISO/IEC $9126 - 1, 2000$) as "the capability of a software product to be used and attractive to the user when used under specified conditions." [84]. It emphasizes the possibility for the users to accomplish the demanded task by using the software with minimum adverse consequences. 3) Efficiency is the ability of an algorithm to perform a task with optimum resources usage. Developing an efficient software requires reduction the number of unnecessary resources used to produce the demanded output [85].

4.4.3 Make it scalable

Scalability is a substantial characteristic of any system. It refers to the competence to cope the increasing in the workload and the operational demands. A scalable software is to endorse any further growth or upgrades without a negative impact on the quality of services and with the minimum incremental costs.

4.5 Application of HPC in EDLD

As it is shown in subsection 2.3.2, three main components, namely; data acquisition, data correction, and data analysis modules can construct a standard EDLD data pipeline. Figure 4.4 visualizes a scheme of the typical EDLD pipeline consisted of the data streaming system (Raccoon), on the one hand, and the analysis system, on the other hand [32, 39]. These two systems work in sequential manner and require network-less data transfer environments (i.e. using an external storage device). This shows several drawbacks, such as time consumption, missing feedback to the user during the experiment, and data loss.
Another disadvantage may arise from software complexity. The absence of a unified framework and utilizing uncoalesced submodules result in implementation complexities, memory issues, and high latency execution time. Moreover, lack of coherency puts more loads on the scientists forcing them to extend their programming skills to fulfill the experience demands of each utilized platform. This can respite the time to discovery, distracting the scientists from their science.

4.5.1 EDLD-tool

EDLD-tool is a new user-friendly GPU-based software package [8] to stream and process datasets taken at X-ray white beam synchrotron sources using an energy-dispersive 2D detector (pnCCD). This tool makes use of many scientific, high performance libraries to increase the precision of data processing and to achieve *on-the-fly*

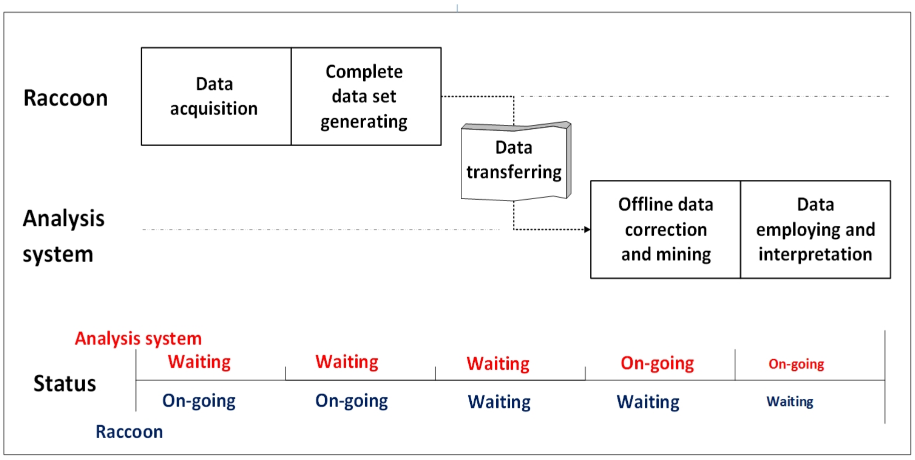


FIGURE 4.4: Present data handling system design. The Raccoon and the analysis system work sequentially. Data are persisted to a storage disk and is transferred manually to the analysis module.

analysis. It aims to find an optimum solution for problems previously addressed, considering the computation efficiency and the precision of data analysis. It is integrated into the readout system to realize efficient data streaming, online visualization, real-time analysis and rapid feedback to the user during the experiment. EDLD-tool has a scalable architecture allowing to scale up its capacity to meet increased workloads. Furthermore, it is designed in a manner that allows to add new functionality without changing the existing code. It based on the abstraction class design pattern with header file shown in appendix B.1. Aiming to achieve the interface segregation principle [86] and to reduce the side effects of required changes, the first release of the tool is divided into five coalesced modules that use the interprocess communication and can run concurrently. Figure 4.5 shows a scheme of the EDLD-tool design which can explained as follows:

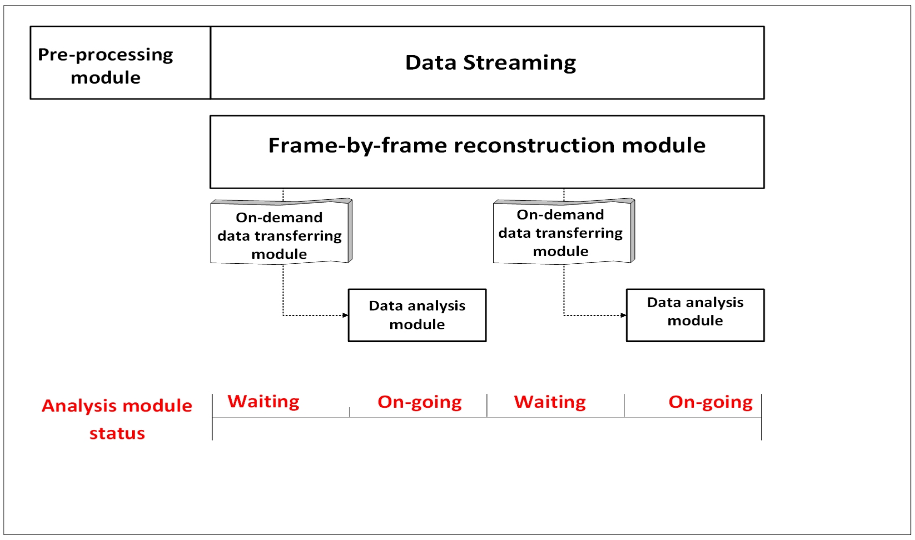


FIGURE 4.5: The EDLD-tool design. The pre-processing module performs its job before starting the experiment. The streaming and reconstruction modules continuously work during the experiment time preventing data loss. The on-demand and analysis modules are mutually exclusive and controlled by the user.

4.5.1.1 Streaming module

It is the data-truck to deliver data from the sources (i.e. the detector readout system in this case or previously recorded frame data files) to the final endpoint(i.e. the execution APIs or the cloud). The streamed datasets are structured based on the Coordinate formate (COO), as explained in subsection 3.5.2, allowing to save datasets in both cloud-based and local-machine storages. It is equipped with a data-allocation function based on the In-Memory Database (IMDB) architecture in which the data are stored in the main physical memory of the local hosting machine. Each pixel is allocated as a structure of data named "*singlePixel*" with header file shown in appendix B.2. This structure contains all details about each individual pixel, such as X-Y position, energy spectrum, and intensity. This function is efficient in case of experiment-steering as it enables the real-time analysis without storing the data. Moreover, in contrast to the tradition disk-optimized database architecture, the IMDB provides considerably high-speed data-access.

4.5.1.2 Pre-processing module

The pre-processing module is designed to perform the detector-scope correction steps required prior to data streaming. These steps (i.e. noise map, offset map , common mode noise, bad pixel map) are executed to deliver the raw pixel data quantitatively allowing to detect individual photons and their actual amplitude. This can be quantified by recording and processing a few hundreds of dark frames in absence of X-rays. This module is based on the technique introduced in [32].

4.5.1.3 Frame-by-frame reconstruction module

The frame-by-frame module is implemented endorsing individual frame treatment. It evaluates and reconstructs each frame event, as explained in section 2.3.3. The reconstructed data are visualized and saved in so-called *'finalized data'* container. Moreover, this module provides the user with many information, such as number of corrected frames, total number of events, and events pattern distribution. The frame-by-frame reconstruction technique is efficient regarding the frame dynamics, as it gives the domain full access to each individual frame-data. This supports further advanced statistical analysis, such as sub-pixel characterization and cluster-analysis [87].

4.5.1.4 On-demand data delivery module

The *'on-demand'* acquisition module optimizes the connection between the streaming module and analysis module. It has a full accessibility to the *'finalized data'* container, allowing the user to control the demanded dataset being analyzed depending on the information provided by the frame-by-frame module. The user-selected dataset is sent to the analysis module.

4.5.1.5 Analysis module

The Analysis tool manages crystallographic parameters of EDLD based on Bragg's law and the Laue diffraction method. It contains two separate workflows: (1) The "*A-Z Automatic*" workflow, which analyses a complete data set automatically based on minimum user intervention. The user only has to provide the main experimental parameters, such as Sample-Detector-Distance, the position of the direct beam on

the detector (X_{ref} and Y_{ref}) and peaks height threshold (see section 2.3). (2) The supervised workflow, in which each analysis-procedure is fully controlled by the user.
The analysis module is able to process previously recorded data file or data received from on-demand module. It can, so far, perform the following main functions:

- Generation of a pixel-map image,
- Generating statistics on hit rate and events patterns distribution,
- Estimation and subtraction of background,
- Identification and localization of Bragg Peaks,
- Analysis of the energy spectra,
- Geometry optimization,
- Parameters calculation (Unit-cell, Bragg norms, reflection angles),
- Auto-indexing of Bragg peaks,
- Grain-corresponding spots identification,
- Calculation of crystal orientation with respect to the laboratory coordinate system.

4.5.2 The Graphical User Interface (GUI)

A Graphical User Interface (GUI) platform is designed to provide the user with familiar working environment. All features and the most frequently used functions are merged in the main control boards, where each feature is connected to a direct manipulation element enabling the user to control the work-flow without any prior technical experience. Figure 4.6a is the streaming module control board where the raw and reconstructed data are visualized. It enables the user to control the cluster size and the frames set that is being analyzed. The control board analysis module is shown in figure 4.6b. It is divided into three parts: 1) the left part is the input boxes, where the user can provide the required parameters. 2) the right part is devoted for energy-wise analysis and visualization. 3) the middle one contains the main image visualization.

4.5.3 EDLD-tool output

The tool is able to display and to save the results of the processed data as an image plot or as a text file. The saved output file is written and customized in readable formats (e.g. pdf, PNG, JPG and text plains), allowing for later data reviewing. The visualized outputs are categorized as following:

4.5.3.1 Frame visualization

Frame visualization enables the user to inspect each individual detector frame. A 3D-plotting scheme, which contains all recorded events assigned in a chosen frame, is generated, as shown in figures 4.7a and 4.7b. Each plot includes a time stamp, which assigns the time difference between the first frame and the plotted one. The

total number of recorded events and the highest assigned event-amplitude are included as well.
The frame visualization provides the basic environment for further advanced frame-scope treatments (e.g. sub-pixel reconstruction, charge cluster analysis). Moreover, the scheduled data visualization technique describes the time-line of the experiment.

4.5.3.2 Comprehensive imaging

A continuous update of the pixel map (figures 4.8a, 4.8b, 4.8c and 4.8d) is generated ensuring a real-time high-quality display of the data which are currently under processing. It is supported by various graphical editing functions (e.g. zooming in/out, color and counting scales changing). Moreover, each localized spot is displayed by 2D and 3D peak intensity plots and X-Y intensity line-profiles distribution (figure 4.8e).

4.5.3.3 Statistical and numerical output

Descriptive statistics on the experiment and the sample (i.e. events patterns distribution, hit rate, indexing of Bragg peaks, crystal orientation, calculation standard deviation, running time) are displayed and/or saved during the execution time.

4.5.4 Technical description of the tool

EDLD-tool is an object-oriented program based on the combination of sequential and parallel computing. It uses the concept of classes inheritance. The current version contains about 50 classes and class libraries (ROOT-CERN [88], OpenCV [89], Eigen [90] and QCustomPlot [91]). The class inheritance has a single-root structure, in which all used classes are inherited from the base class. The software package is written in C/C++ and CUDA [92] with a user interface designed in Qt cross-platform [93].

4.5.4.1 Streaming module

It is operating in non-blocking mode based on the ring-buffering technique. Architecturally, it is inherited from the algorithm in [39]. It is connected to the readout hardware, allowing a frame streaming rate up to 600 frame/sec. However, due to the beam flux constraints in EDLD experiments, the maximum frame rate is 100 frame/sec gives a single frame streaming cycle time = 10 msec.

4.5.4.2 Frame-by-frame reconstruction module

The base routine (CPU-wise) pseudo-code is shown in algorithm 1. A single data frame is loaded from the streaming module and is treated as a 2D matrix with a size equal to the detector sensitive area (384×384 in our case). Subsequently, the reconstructed frame data are being visualized in real-time. The frame reconstruction-visualization cycle time for the currently used machine is 1.1 ± 0.1 sec. However, using a sequential frame-by-frame treatment routine with a high rate readout may

induce a high latency execution time, data loss and online visualization lag. Therefore, a GPU-based routine has been developed in order to perform parallel reconstruction. A set of frames data is vectorized and sent to the GPU machine, whereby each frame is treated by a single thread. This attains a speedup factor up to 14 for the given system features in table 4.1.
In order to optimize the number of frames being reconstructed in parallel per cycle, a timing analysis considering the Worst-Case Scenarios (WCSs) has been performed. The WCS was assumed to be a single frame streaming cycle time = 10 msec and reconstruction-visualization cycle time per thread = 1.2 sec. The output parameters for the WCS analysis are demonstrated in table 4.2. As a result, the value of 200 frames per cycle presents the optimum choice, as it obtains an acceptable speedup factor, short waiting time of the analysis module, no data-loss and short real-time visualization lag. The pseudo-code of the used routine is displayed in algorithm 2.

Algorithm 1 Reconstruction base algorithm.

```
1: function DEFINE FRAME DATA MATRIX
2:     frm[384][384]← streamed frame data
3: function MAIN FUNCTION
4:     FindValidPatterns;
5:     ReconstructAllPatterns:
6:     Singles ← Number single events;
7:     Multi ← Number multipixel events;
8:     SaveReconstructedData;
9:     UpdateRealTimeVisualization;
10: startNewFram
```

TABLE 4.1: The used system specifications.

Feature	Specification
Model	Dell Lattitude 5480
Processor	Intel Core i7-7600U
Speed	2.8GHz(4 CPU)
GPU	Nvidia GeForce 930MX

TABLE 4.2: System timing analysis.

Number of frames	Speedup factor	First parallel reconstruction cycle time [sec]	Further parallel reconstruction cycle time [sec]	Analysis module waiting time [sec]/cycle	Real-time visualization updating frequency [Hz]	Average data loss/cycle
100	3	2.2	1.2	0	1	20 frame
200	6	3.2	1.2	0.8	0.5	0
500	12	5.2	1.2	3.8	0.2	0
1000	12	12.4	2.4	7.6	0.1	0

4.5.4.3 Analysis module

The main tasks of the analysis module are to reduce the data size, to orient the data-pattern towards the desired application and to provide physical meaningful conclusions from the processed data. It is divided into three submodules.

Algorithm 2 Reconstruction GPU-based algorithm.

```
1: function THE KERNELFUNCTION
2:     the reconstruction base algorithm          ▷ Without Updating the real-time visualization
3: procedure INITIALIZATION
4:     checking function =false
5:     Num = 0
6:     i = 1
7: function CHECKING FUNCTION
8:     while (false) do
9:         Num ← Number of streamed frames
10:        if Num > (200*i) then return true
11: function MAIN FUNCTION
12:     if (true) then
13:         DefineNumberOfThreads;
14:         CopyToDevice;
15:         LaunchKernelFunction (NumberOfThreads);
16:         syncThreads;
17:         CopyToHost;
18:         UpdateRealTimeVisualization;
19:     i ← i+1
20:     checking function ← false
21: goto checking function.
```

1) Data mining and exclusion of artifacts The data mining routine is equipped with several functions, such as the calculation of an absolute intensity map, direct-beam discrimination, background elimination, Laue spot localization and energies determination. To elaborate data mining, we undergo the pnCCD-data to the following steps:

- Pixels intensity map calculation: In the single photon detection mode, each pixel will count one or no photon per frame. The integrated image, known as pixels intensity map, is the summation of all frames providing the number of events recorded by each pixel during the whole recording time. To avoid pixel-by-pixel sequential iteration, a GPU-based algorithm is developed. Using the algorithms in [94] and [95] as the main conceptional references, the integrated image is generated with speedup factor up to 5.

- Background elimination: Scattering from experimental setup and electronic noise may produce a continuous background. An effective method is needed to detect and subtract this useless signals. Depending on the statistics-sensitive non-linear iterative peak-clipping (SNIP) algorithm developed by CERN [96], the 2D-continuous background is separated. The reliability of SNIP algorithm to identify the background for a Laue experiment has been tested and reported in [97, 98].

- Data delivery: Once the pixel map is corrected, it is delivered for further processing. The rest of the data concerning the events amplitudes is switched to a holding-on phase. In other words, the energy information for each recorded event is stored and only the position information is delivered at this moment.

- Laue spots (foreground) localization: This is achieved by applying a two-dimensional high-resolution peak search function [99] on the delivered pixel map. This function automatically identifies the peaks in an image based on deconvolution method. A detailed description of this function concerning the mechanism and the test of reliability can be found in [100, 101]. As soon as this step is carried out, a 2D-Gaussian fitting is applied on the localized peaks to find the center of mass for each spot with precision $\approx 10^{-2}$ pixels. A list of localized spots with their positions is delivered to the next step.

- Energy spectrum estimation: Using the delivered spots list and data in holding-on phase, an energy histogram is generated for each Laue spot representing the distribution of the event amplitude in this area. To extract the required energy-information from the histogram, a 1D spectrum peak searching function is used. This function selects the main peak after performing background-elimination, spectrum smoothing and Compton edge recognition[99, 102]. Hence, the selected peak is subjected to a 1D-Gaussian fitting.

Finally, the data is compromised to be a list of localized Laue spots with their energy values. Only this list is sent to further process.

2) Data interpretation Many numerical calculations and crystallographic relations are implemented at this step aiming to provide physical meaningful conclusions from the processed data. In order to sustain with the maximum computational performance, many parallel computation routines are implemented:

- Calculating the required parameters: Many parameters (e.g. the diffraction angle 2θ, the wavelength λ, possible Miller indexations list (h, k, l) etc) for each localized spot are calculated. Accordingly, as the number of localized spots increases, the processing time increases as well. Therefore, a GPU-based routine has been implemented for crowded-spot patterns, allowing for a speedup factor up to 5. Considering an efficiency analysis of the achieved parallelism, the GPU routine is implemented only if more than 30 reflections have been identified.

- Peak auto-indexation: To find the correct indexation of a Laue spot, one compares the angles between two measured reflections with the theoretic angles for all the possible permutations of Miller indices. If the difference between experimental and expected angles for a given crystal system fulfills a predefined uncertainty, the indices are supposed to be correct[103].
  This base serial routine requires $(n \times m)^2$ action to perform a full indexation task, where n is the number of reflections and m is the number of the possible h,k,l for the nth reflection. A GPU-wise routine is implemented to parallelize the indexation step reducing the work complexity to be $(n \times m)$. A further parallelization implementation for better work complexity optimization is still under development.

- Grain-corresponding spots identification: To assemble the collected reflections into the correct corresponding grain is to compare the angles between two measured reflections with the theoretic angles for all the possible permutations of Miller indices. If the difference between experimental and expected angles for a given crystal system fulfills a predefined uncertainty, the couple is supposed to be generated by planes belong to the same grain. A detailed

explanation of this procedure is given in Appendix C.
The base serial routine has a time complexity of $!n$ (i.e. $O(!n)$) to perform a full identification task, where n is the number of reflections. A GPU-wise routine is implemented to parallelize the indexation step aiming to reduce the work complexity. A further parallelization implementation for better work complexity optimization is still under development.

- Orientation determination: A transformation matrix is applied to transform any plane in a crystal from crystal coordinates to Cartesian coordinates [104]. It can be calculated with only three reflections [103]. For crystals having more than three reflections, the reflections are grouped in threefold subgroups covering all possible mutations. The final orientation is the mean value of the subgroups orientations. This step is done by the space transformation modules implemented in Eigen library.

- Geometry optimization: EDLD-tool is equipped with a supplementary function that deals with geometrical parameters, such as SSD, peak position, and direct beam position as ranged value with limits and step size defined by the user. The data interpretation procedure is performed, covering all permutations of these values within the predefined range. Only discrete values that show the lowest standard deviation are voted for the further calculations. The main goal of this function is to find the optimal values of the experimental setups.

3) Data visualization Data visualization establishes the connection channels between the user and the data being processed. EDLD-Tool is supporting this by many cross-platform application programming interfaces (APIs) for rendering 2D and 3D vector graphics, such as Open Graphic Library (OpenGl), QCustomPlot widget and OpenCV. That endorses the interaction with a GPU to achieve hardware-accelerated rendering.

4.6 Results and discussion

A previously conducted experiment [105] has been utilized to evaluate the performance of the new analysis system. This experiment was realized at the synchrotron radiation facility (EDDI beamline, BESSY-II, Berlin) with white X-rays. A Gallium-Arsenide (GaAs) sample was located between the beam slit system and the detector active area with experimental parameters given in table 4.3. The raw data were streamed by streaming-from-file module and reconstructed by the parallel frame-by-frame module. The finalized data were transferred and processed by EDLD-tool analysis module ans a full analysis procedure was performed.
Table 4.4 reports the results and the comparison between both approaches. Notably, EDLD-tool shows almost the same results as the previous approach regarding the accuracy. The lower number of localized Laue spots may be a result of low intensity peak height less than the predefined threshold or statistical error out of the accetable range. Utilizing of the GPU-based algorithm enables the EDLD-tool to execute the event-reconstruction within a time scale of minutes, comparing with the traditional method that requires several hours to execute the same task. In respect of data analysis and interpretation, EDLD-tool shows an outstanding performance as it can execute a full data set analysis within a few seconds. The Geometry parameters can be

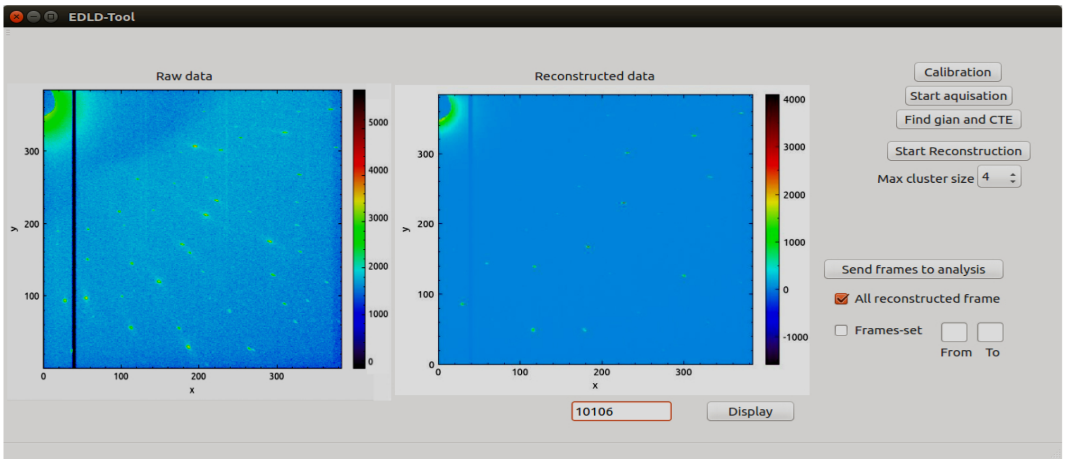
exclusively optimized by using the "Geometry optimization function", aiming to reduce the statistical error. This function is not available in the traditional techniques. However, activating this function may increase the latency and the workload. The pioneering feature of the EDLD-tool is that it can operate "on-the-fly", providing the user with real-time report about the conducting experiment.

TABLE 4.3: Experimental setup.

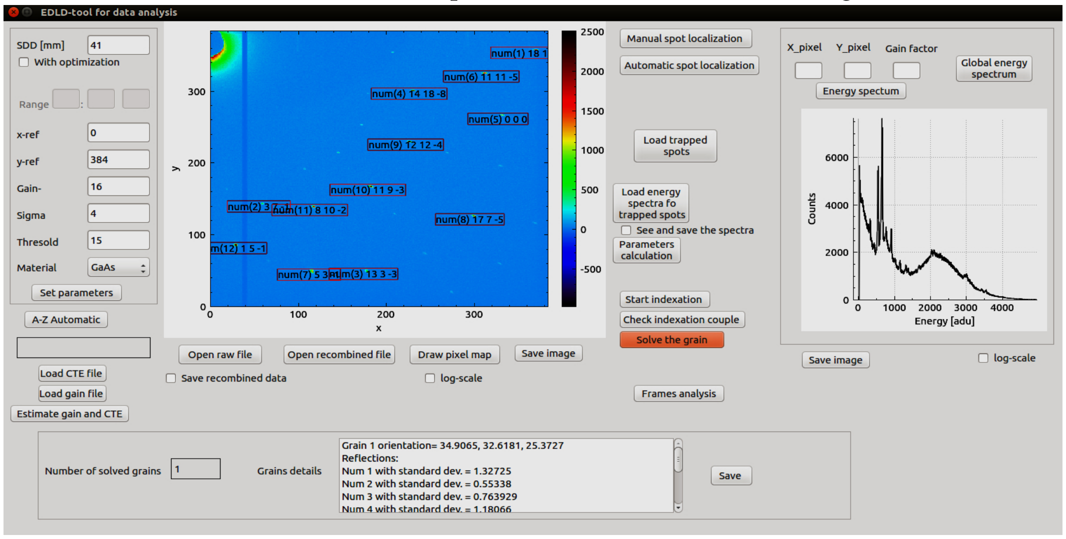
Material	GaAs
Lattice constant	5.653 Å
Beam	White x-ray beam
Beam energy	40 : 140 keV
Beam size	100*100±5 μm
Absorbing layer	200 μm of Pb

TABLE 4.4: Results comparison between the previous analysis system and EDLD-tool.

	Previous analysis	**EDLD-tool**
Required disk-storage	20 GBytes	0
Computing style	off-line	on-the-fly
Work flow control	Manually done by the user	Automatically
Elapsed time for reconstruction	Few hours	Few minutes
Elapsed time for analysis	Few hours	Few seconds
Number of reconstructed frames	100000	100000
Geometry optimization	Not included	Exclusively performed
Number of localized spots	100	98
Number of indexed spots	100	98
Parameters calculation accuracy	0.5%	0.5%
Indexation standard deviation	2%	2%



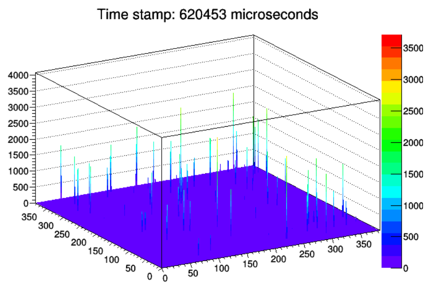
(A) EDLD-tool streaming module control board. The left visualization box shows the raw data streamed directly from the readout. The right one is the data after reconstructing process with a counter to display the number of finalized frames. Each function and feature is connected with it manipulation element located in the right.



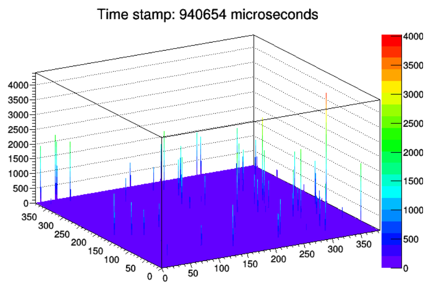
```
Grain 1 orientation= 34.9065, 32.6181, 25.3727
Reflections:
Num 1 with standard dev. = 1.32725
Num 2 with standard dev. = 0.55338
Num 3 with standard dev. = 0.763929
Num 4 with standard dev. = 1.18066
```

(B) EDLD-tool data analysis module control board. The lift-side part is for the user input parameters and commands. The middle part contains all analysis main functions and pixel-map visualization. The right-side part is for energy-scope analysis. The lower information palette displays the final results.

FIGURE 4.6: EDLD-Tool graphical user interface main boards.

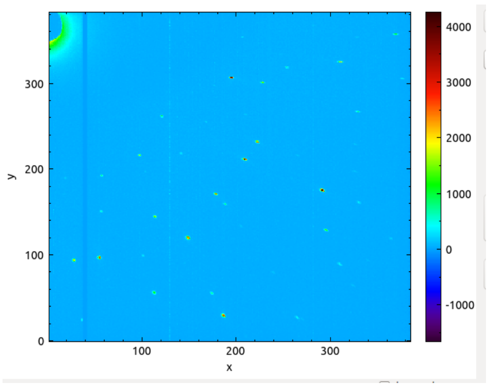


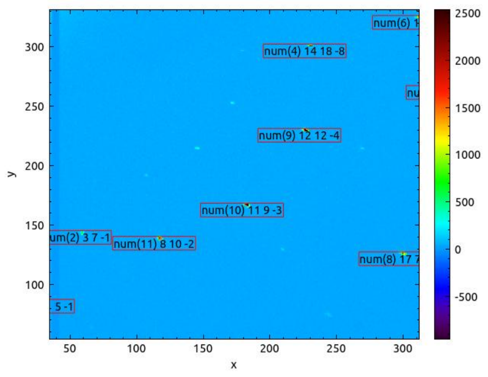
(A)



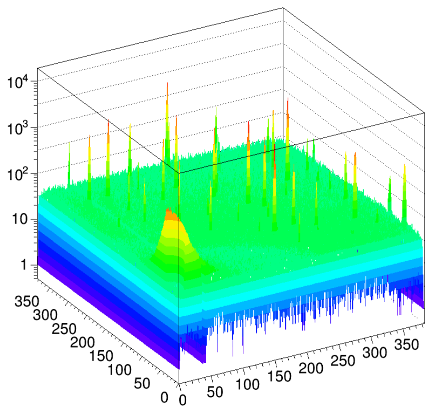
(B)

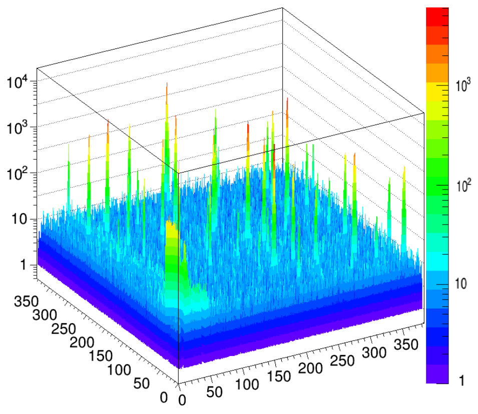
FIGURE 4.7: Frame visualization output. (A) is a 3D view of frame number 400 and (B) is for frame 500. The vertical axes represent the event amplitude in [adu]

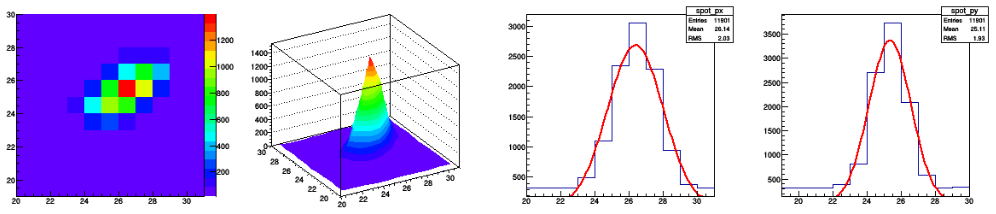

(A) A screen-shot of the integrated image after data reconstruction



(B) A screen-shot of a part of the integrated imaged after the analysis procedure


(C) 3D plotting of the pixel map before background elimination


(D) 3D plotting of the pixel map after background subtraction



(E) The quaternary plotting output of a localized Laue spot

FIGURE 4.8: The visualized output of EDLD-Tool for a testing Gallium-Arsenide (GaAs) sample exposed to hard white X-ray beam with energy range from 5 keV up to 120 keV.

Chapter 5

Artificial Intelligence (AI)

"Some people call this artificial intelligence, but the reality is this technology will enhance us. So instead of artificial intelligence, I think we'll augment our intelligence."- Ginni Rometty

5.1 Introduction

The term *"intelligence"* has been a subject of controversy throughout the history of psychology. There is no uniform definition of the term: Versatile researchers have proposed different conceptualizations which suggest that intelligence is the ability to learn as well as recognize and solve problems. It involves many aspects such as awareness, understanding, memory, language, and planning [106]. Thus, the following questions arise : Is intelligence an exclusive property of the human brain? Can we create intelligent machines that have the same level of intelligence as humans? Alan Turing , who is considered to be the father of artificial intelligence (AI) and computer science, initiated this branch of research by inventing the first intelligent machine, the Turing machine, in 1936 [107]. This machine was used during the Second World War to break the "Enigma" code used by German forces to send messages securely. In 1943, the field of AI has been fostered by Warren McCulloch, who created the first neural network model [108]. . Arthur Lee Samuel also contributed to the establishment of AI as well as computer gaming by publishing a paper about his novel approach , which resulted in the first self-learning program to play checkers [109]. Since then, a lot of other attempts has been made for the advancement of AI, which overall aim at making the dream to create smart machines that mimic the human behavior come true.
Succinctly speaking, AI is the scientific field which aims to study and design intelligent systems that perceive the environment and make decisions to maximize the probability of success.Generally, AI can be classified into three types [110]:

1. Narrow AI: This type of intelligence is prominent in focusing on a single narrow task, such as playing chess, marketing suggestions, and weather forecasts.
2. Strong AI: : This type of intelligence is intended to think and perform several tasks , similar to a human being. Moreover, it is able to learn, criticize, and develop itself.
3. Super AI: This type of intelligence is capable to perform more sophisticated actions, which are beyond the capacities of human intelligence in many fields, such as scientific creativity, and social skills.

Moreover, AI is incorporated into a variety of disciplines, such as automation, automotive (self -driving cars), robotics, natural language processing (NLP), machine vision, and machine learning (ML), as figure 5.1 demonstrates. The focus of this work is on ML, which is highlighted in green. As can be seen, ML contains Deep Learning (DL)technology as a subset. This chapter shows how implementation of AI might help to develop incredibly exciting and powerful techniques to solve many scientific problems. Section 5.2 presents a collective summery of the ML fundamentals and exemplifies its application in the crystallography community, whereas section 5.3 then zooms in on the subfield of ML, namely DL, by outlining its basics and instantiating its application in the crystallography community.

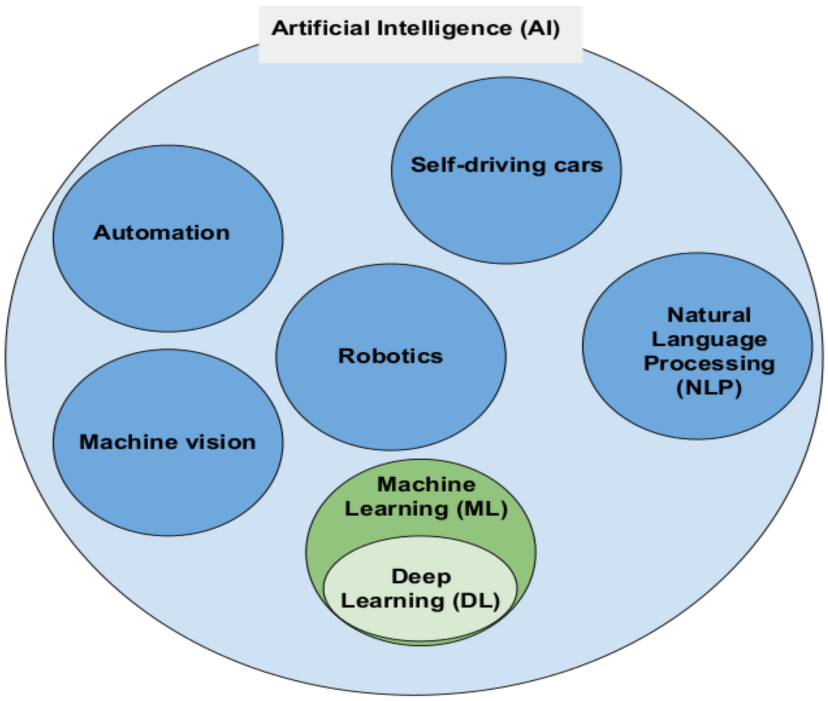


FIGURE 5.1: AI technologies.

5.2 Machine Learning (ML)

As stated in the introduction, ML is a form of AI that exploits datasets to detect a pattern and learn it. It is one of the most important techniques to leverage data, helping organizations to enhance their level of awareness and understanding. Compared with Classical computer programs, the usage of such algorithms can automate task performance rather than explicit programming. In industrial sector, as an example, enterprises have exploited variety of ML models to boost the capability of predicting the market fluctuations. This helps the decision-makers to create better future for their business. According to Oxford Economics survey in 2017 [111], approximately 50% of companies are already using innovative ML techniques for business analysis, such as repetitive tasks automation, recommendation systems, and data pattern recognition.
In this manner, the science has acquired its AI-portion. The effect of ML is clearly

visible across empirical sciences (i.e. biology, cosmology, material science, social science, and different subfields of engineering). Many scientific applications and research works have been published, broadly covering a range of scientific concerns, such as bioinformatics [112, 113], medicine [114, 115], and astronomy [116, 117].
This subsection is structured as follows: Subsection 5.2.1 provides an overview of the disciplines involved. Next, subsection 5.2.2 distinguishes a typology of learning, while subsection 5.2.3 lists the most common algorithms. Finally, subsection 5.2.4 demonstrates an application of ML to solve a crystallographic problem, namely grain-corresponding Laue spots classification in a polycrystalline ED Laue pattern. The detailed overview of ML concepts and theories can be found in [118–120].

5.2.1 Disciplines

" Machine Learning in a natural outgrowth of the intersection of computer science and statistics " [121]. Professor Tom Mitchell introduced ML as a joint point where different disciplines meet. He suggested that statics and statistical learning theory are the backbone of any intelligent algorithm. He also state that ML incorporates additional data-wise disciplines, such as data capturing, storage, indexing, retrieval, and merging. At a later stage, neurocomputimg and pattern recognition have become other foci this field. All in all, it can be summarized that ML is a subject that integrates a few widely mentioned disciplines, namely data science, statistics, neurocomputing, and pattern recognition. Figure 5.2 depicts these four main areas involved in the solution of a learning problem.

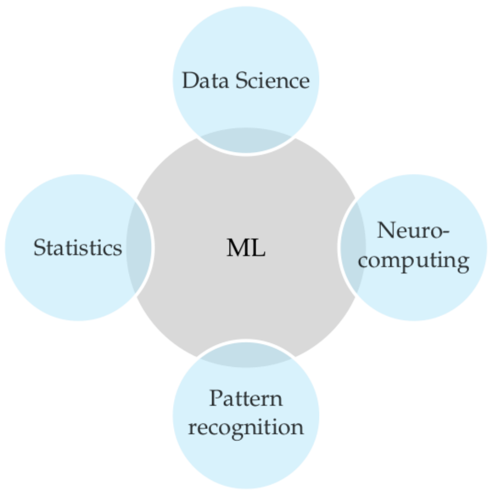


FIGURE 5.2: Machine learning disciplines

5.2.2 Types of learning

It is essential to identify the types of learning and to describe how they exhibit themselves in any given task one may encounter. Understanding the types of machine learning helps the developer to craft the proper learning environment. Commonly, learning might be divided into several categories according to its dependency on data. Accordingly, as shown in figure 5.3, the three main categories are the following: 1) *supervised learning*, where an algorithm learns from labeled data and predicts the next *"never-seen-before"*value [122], 2) *unsupervised learning*, where an agent identifies clusters from unlabeled data [123], 3) *reinforcement learning*, where an agent learns from mistakes by interacting with a changing environment [124].

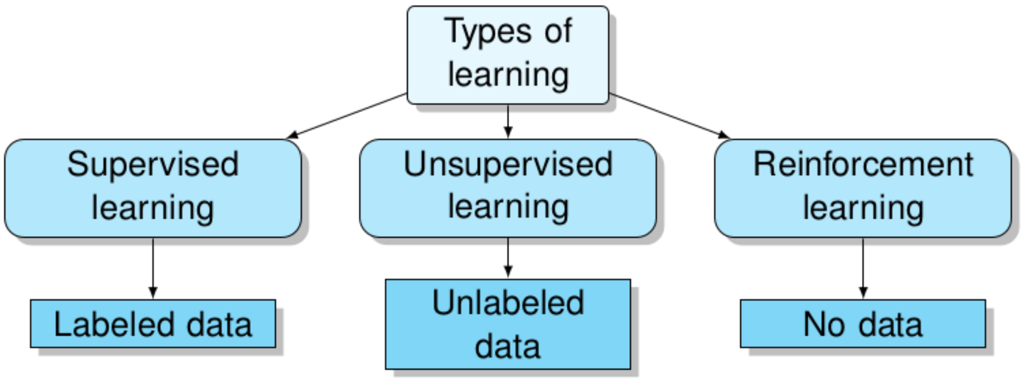


FIGURE 5.3: Different types of learning according to their dependency on data

5.2.2.1 Supervised learning

Supervised learning is the most popular paradigm, in which a learner algorithm learns from a given dataset in the form of examples with labels. The algorithm is fed with two datasets, so called *"the training and the testing sets"*. The training set consists of n ordered instances of (x_i, y_i) pairs, where x_i and y_i are the i^{th} input and output respectively [125]. Due to a sufficient amount of such labeled (i.e with the known output) sets, the algorithm can identify unlabeled data with very high accuracy. During training phase, the testing dataset is used to evaluate the learner. First, it is sent to the learner as an unlabeled set, allowing the algorithm to predict the label for each example. Then, a comparison between the predicted and the actual output is performed in order to estimate the accuracy of the algorithm.
Practically, a supervised learner tries to correlate the input features with the target variable (i.e. the prediction output), generating a prediction function used to map new input to the final output. This is a learning type which is exhibited in many applications, such as spam detection, face recognition, handwriting recognition, and advertisement popularity.

5.2.2.2 Unsupervised learning

Unsupervised learning is the opposite of supervised learning, where a learner is fed by unlabeled datasets. In other words, the algorithm features no output or target outcome to predict. Instead, it tries to cluster the input population in different groups, categories, or classes in such a way that makes sense for further intervention. Nowadays, unsupervised learning is a demanding area as the overwhelming majority of data is unlabeled. Such algorithms enable the user to handle huge amount of data, achieving all potential profit out of it. They are widely used in several areas, such as grouping user logs, recommender systems, and buying habits.

5.2.2.3 Reinforcement learning

In this case, machines or software agents are trained to make specific decisions by using reinforcement learning. Conceptually, a learner is exposed to a dynamically changing environment and trained to determine the ideal behavior to boost its performance. It has a *learning-from-mistakes* approach, where the learner receives reward or reinforcement signal from the environment after performing an action. As long as this process is repeated, the learner is able to distinguish between bad and good behavior. Subsequently, the decision-making mechanism of the agent is improved. This type of learning is used in many industries, such as resource management, industrial simulation, and video games.

5.2.3 Common ML algorithms

This part gives a brief overview on the most common machine learning algorithms. These algorithms can be used to solve a huge range of problems. More applications can be found in [119, 120].

5.2.3.1 Linear regression

Regression concept is originated from statistics has been adapted to statistical machine learning. Linear regression is used to model the linearity between independent and dependent variables. This can be done by means of linear fitting by applying the following line equation:

$$Y = \beta_0 X + \beta_1 \tag{5.1}$$

where Y is a dependent variable, X is an independent variable, β_0 represents the slope, and β_1 is the intercept.
Linear regression is mainly classified in two groups, as follows:

- simple linear regression, which is modeled by only one independent variable.
- multiple linear regression, which is modeled by more than one independent variables.

Linear regression is co-opted in a wide range of scientific areas (e.g. biology, epidemiology, and social science) and applications (e.g. trend-line estimation, fixed investment)

5.2.3.2 Logistic regression

Logistic regression is considered as a classification algorithm, since it is used to model dichotomous values (i.e. 0/1, yes/no, true/false). It is known also as the logistic model or logit model. It is used to analyze the relationship between multiple independent variables and a discrete dependent variable. It is based on the occurrence-probability estimation of an event, where the dataset is fitted and represented on a logistic curve [126].
Suppose the two binary value 0 and 1 to be the outcomes with their probabilities of observation $1 - p$ and p respectively. Then the corresponding logit (l) transformation is written as:

$$l = logit(p) = \ln(\frac{p}{1-p}) = \beta_0 X + \beta_1 \tag{5.2}$$

where the ration $(\frac{p}{1-p})$ is called *odds* and *logit* is the logarithm of the odds [127].
Approximately 70% of data-wise problems are logistic and classification problems. This makes logistic regression one of the most popular prediction methods, which is used in many applications, such as image segmentation and categorization, cancer detection, handwriting recognition, and gender detection.

5.2.3.3 Naive Bayes (NB)

NB is an intuitive probabilistic algorithm used for classification problems. The main idea behind it is to apply Bayes theorem [128] to discriminate different objects based on certain features, following equation 5.3.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{5.3}$$

where:

- A and B are two independent events having positive probability
- $P(A|B)$ is the posterior probability of A occurring under assumption that B has occurred
- $P(B|A)$ is the likelihood probability of B occurring under assumption that B has occurred
- $P(A)$ is the class prior probability of A occurring
- $P(B)$ is the predictor prior probability of B occurring

Accordingly, a NB classifier predicts the probability of belonging to a particular class for given data points. NB algorithms are used in many applications, such as emotion analysis, Spam filtering, and recommendation systems.

5.2.3.4 Random Forest (RF)

It is a classification and regression ML algorithm based on the decision tree concept [129]. It consists of a combination of many uncorrelated decision trees considered as fundamental building blocks of a RF. Each tree individually operates as a separated classifier module. Then, the RF gathers all classifications and votes for the optimal decision. Often, RF algorithms are called *"the algorithms of wisdom of the crowds"*, as the trees act as members of the committee [130]. Therefore, lower correlation between trees gives high performance and accuracy. Figure 5.4 visualizes a scheme of a RF model which has five decision members. Four of them vote for *"True"* while only one is for *"False"*. The RF final decision follows the majority of votes; it has to be *"True"*. RF models are very efficient on large datasets and are able to handle missing data. They are used in many sectors, such as banking sector, medicines, and E-commerce.

5.2.3.5 K- Nearest Neighbour (KNN)

KNN is a non-parametric learning algorithm, where underlying data points are classified into several groups, so-called clusters. A learner then predicts a corresponding cluster for a new data point based on feature similarity. KNN is efficient with respect to accuracy, simplicity, and versatility. However, it is computationally expensive and requires high memory.

5.2.3.6 K-means

K-means is a simple unsupervised algorithm that aggregates each homogeneous data points group in one cluster. Each cluster has a specific centroid, which is a

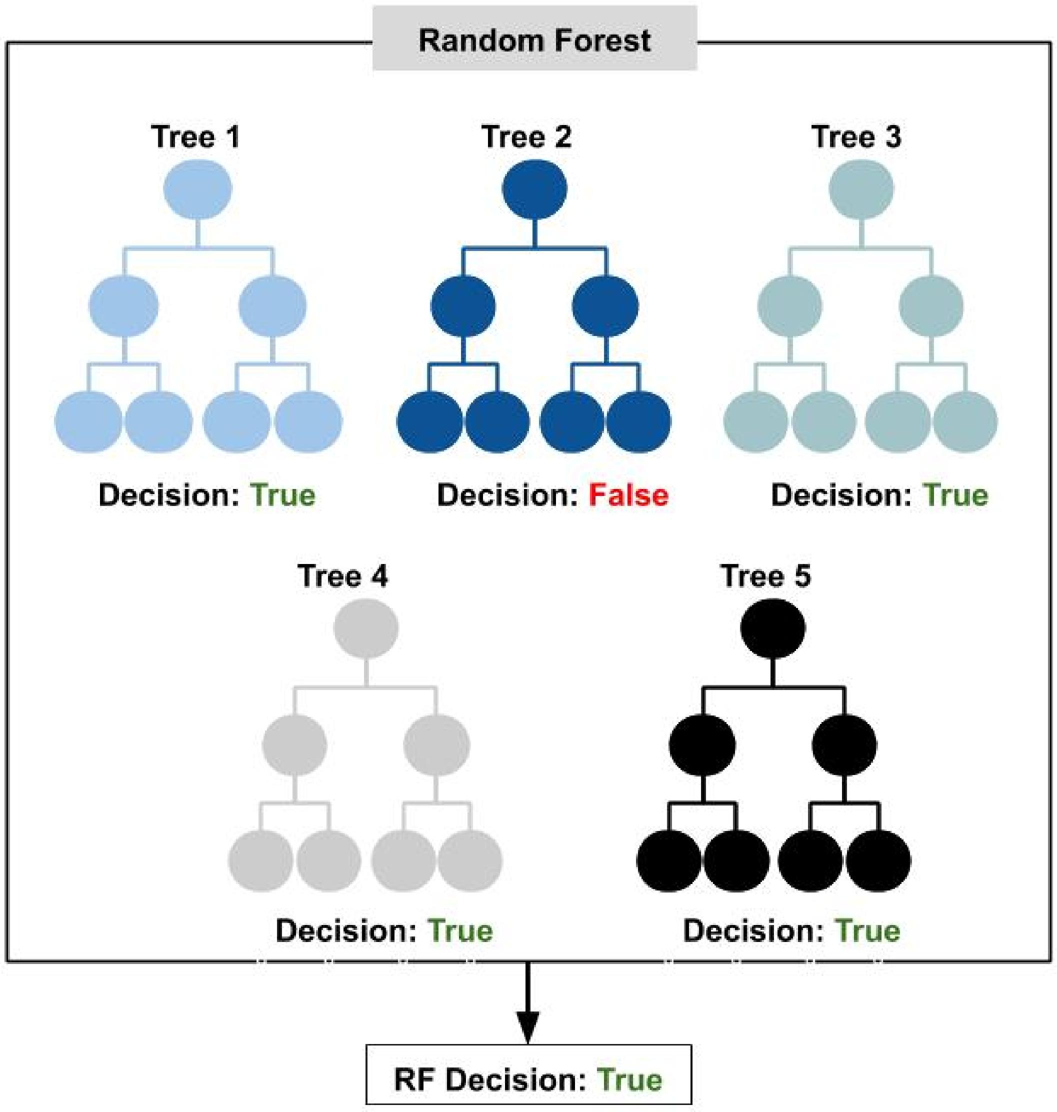


FIGURE 5.4: The RF model for 5 decision trees. The winner is True with the ratio of 80%.

point at the center of a cluster. Then, it identifies the number of centroids *k* (i.e. the number of clusters), fulfilling the fact that data points in a cluster are heterogeneous to peer groups.
The traditional k-mean has a major shortcoming; the domain should provide the number of clusters (*k*) within a dataset. Therefore, many algorithms and techniques have been developed to propose a solution for such this deficiency. This part covers only one algorithm (hierarchical clustering) and one technique (elbow method) which are used for further work. More algorithms and techniques can be found in [131–133].

a) Hierarchical Clustering (HC):
It is a clustering algorithm that considers *k* as a learning parameter. First, it treats each data point as an individual cluster. Then, each cluster is merged to the nearest one. This procedure is repeated till all data points are merged in one cluster and organized as a tree. Each cluster in the tree is called *a leaf* and represents a union of its children clusters as illustrated in figure 5.5. The figure shows a HC pipeline for

an imaginary dataset, which contains eight data points. At the outset, each point is a singleton cluster and then successively agglomerate pairs of nearest clusters. Finally, all points are consolidated into one nested cluster. This type of clustering is typically visualized as a dendrogram, as shown in figure 5.6. The data objects are represented on the horizontal axis. The vertical line gives the average distance measured between the merged points (i.e. the similarity degree of clusters). The red doted line is the threshold representing the demanded maximum error defined by the domain. The clusters are allocated by the number the intersection points between the threshold line and the dendrogram tree.

b) The Elbow method:
It is an ambiguous method to validate the consistency of a clustering algorithm. The idea of the Elbow method is to perform k-means clustering with a range of values for k, calculating the sum of squared errors (SSE) for each value of k. The total SSE indicates the compactness of the clustering and should be minimal. By examining the SSE convergence, the number of clusters is chosen so that the effect of merging an extra cluster onto the data model is ignorable. Figure 5.7 provides an example of the Elbow method output diagram. It shows a range of *k* between 1 and 8 clusters with a chosen Elbow point of 4 clusters.

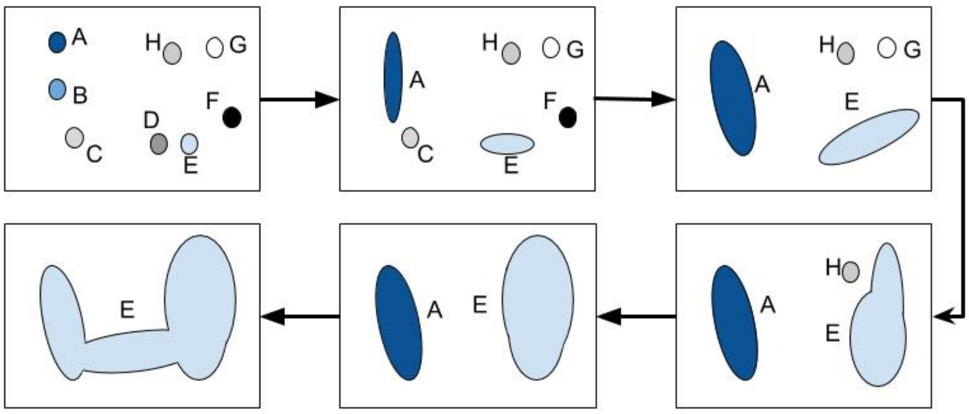


FIGURE 5.5: The HC pipeline for a dataset of 8 objects. At the beginning all objects are represented in individual clusters. Then, they are gradually merged into on cluster.

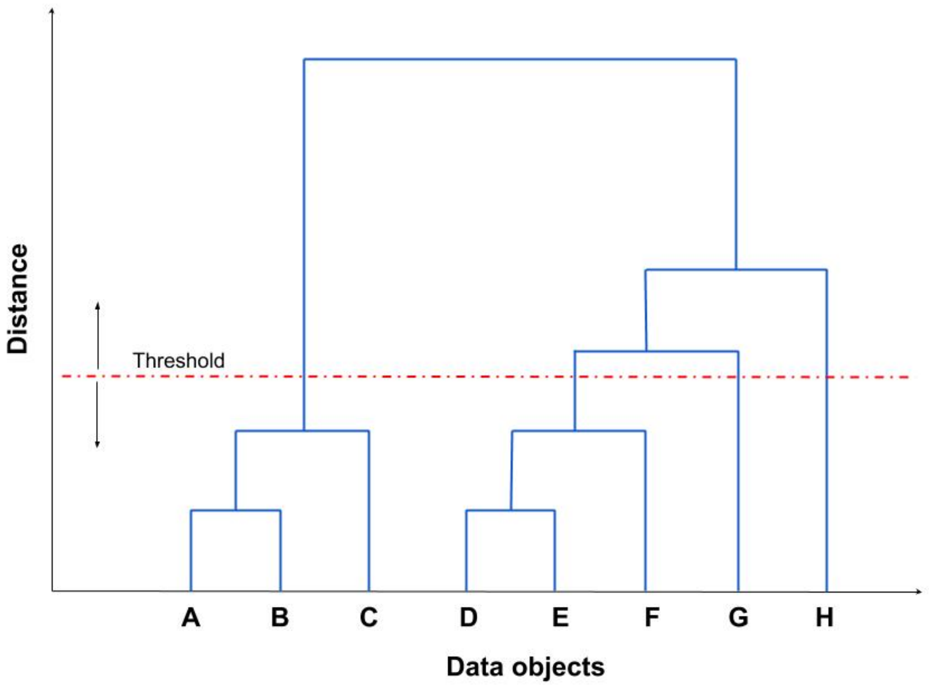


FIGURE 5.6: The output dendogram of the HC.

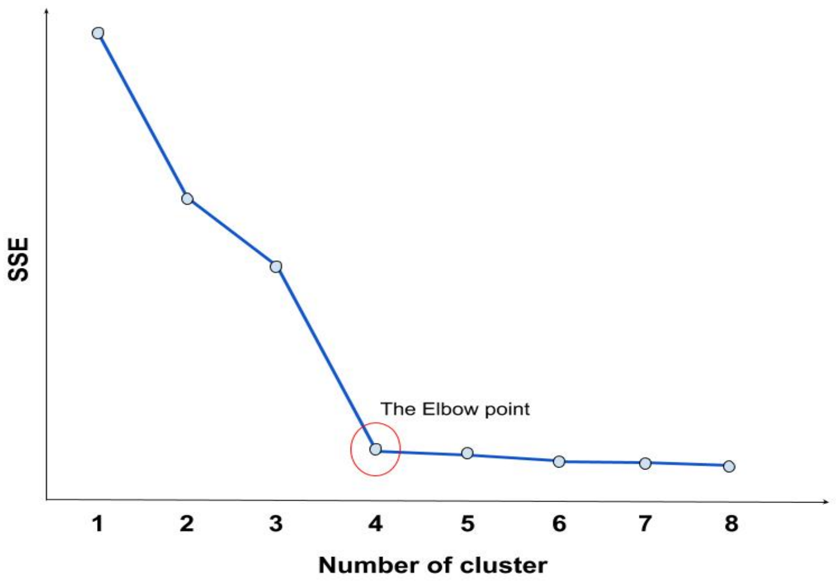


FIGURE 5.7: The output diagram of the Elbow method application with k range between 1 : 8 clusters. The chosen number of clusters is 4, representing the Elbow point.

5.2.4 Application of ML in the grain-related classification of Laue spots in a polycrystalline ED Laue pattern

As it explained in chapter 2, one of the most pioneering advantages of EDLD is the one-shot experiment for polycrystalline materials. However, real-time analysis of the generated Laue pattern without any orientation preference requires innovative techniques to extract grain-wise information. Although, as proven in chapter **??**, HPC is an alternative candidate to perform such analysis, it demands a specific level of programming skills. This part shows how an AI algorithm is able to execute the same task without any programming complexities.

5.2.4.1 The objective

The main aim is to solve a crystallography problem, namely grain-related classification of Laue spots in a polycrystalline ED Laue pattern. This can be considered as a clustering problem, where spots corresponding to one grain representing a cluster and number of centroids (k) refers to number of probed grains. Next step is to find a similarity feature to base the clustering on. Since crystallographic planes within the same crystal have the same orientation degree with respect to a reference crystal, the orientation angle of the planes within a grain to their twine reference planes is considered as the learning feature. Simply put, figure 5.8 visualizes two grains, where *"Grain 2"* is oriented to the reference *"Grain 1"*. Two identical planes $(hkl)_1$ and $(hkl)_2$ are displayed in both grains. The n_1 and n_2 vectors are the norm of the planes in *"Grain 1"*, while n_3 and n_4 belong to *"Grain 2"*. From this perspective, *"Grain 2"* is nothing but a representation of *"Grain 1"* in a different coordinate system. Thus, it might be concluded that the angles α and β represent the axis transformation angle and are identical, following equation 5.4 .

$$\epsilon_2 = \cos\alpha = \cos\beta = \frac{\vec{n}_1.\vec{n}_4}{||\vec{n}_1||||\vec{n}_4||} = \frac{\vec{n}_2.\vec{n}_3}{||\vec{n}_2||||\vec{n}_3||} \tag{5.4}$$

where ϵ_2 is the orientation stamp for *"Grain 2"*. Then, this 3D angle is projected onto $x-y$, $y-z$, and $z-x$ planes, giving the three components ϵ_x, ϵ_y, and ϵ_z. However, this fact requires two extra steps to prepare the data points before feeding the learner. First, finding the reference grain. Second, representing all reflections in the orientation stamp space, so-called *"AT-map"*.

5.2.4.2 Finding the reference grain

The reference grain should contain all possible reflections for a specific experimental setup. That requires a detector active area covering a solid angle of a hemisphere with the radius which equals to the traveling distance between the sample and the detector. The limited size of the detector utilized by an EDLD experiment prevents to fulfill this criterion. Therefore, a possible solution is to simulate the reference grain as an extra step for each dataset.
A special algorithm has been developed to realize this step. It is an inverse modeling algorithm based on projection engineering and vectors algebra, which is written in Python platform. It is able to model a list of all possible reflections generated by a given crystal structure for specific experimental conditions (e.g. detector position to the primary, energy range, and Miller indices range). This is done by generating the reciprocal space map for a bulk crystal (i.e. which has no orientation to the lab coordinates), as follows:

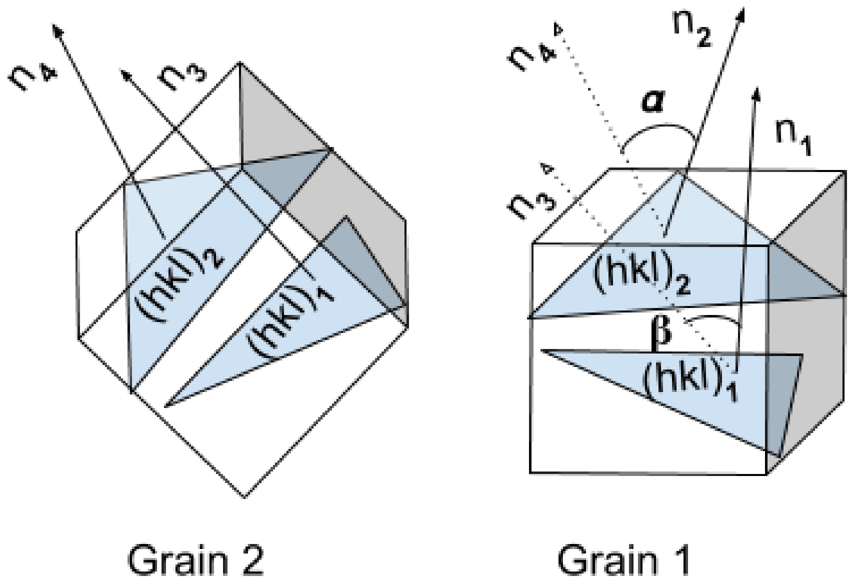


FIGURE 5.8: ML scheme.

A list of possible Miller indices is generated within the predefined range after considering the structure factor of the material under inspection. As soon as the *HKL* list is ready, the primitive basis vectors $\vec{b_1}$, $\vec{b_1}$, and $\vec{b_1}$ for each Miller indices group is calculated according to equation C.4. Applying the equations **??**, C.6, and C.9 the scattering vector components ($\vec{q_x}$, $\vec{q_y}$, and $\vec{q_z}$) are then calculated, which are the main components of the reciprocal mapping. Figure 5.9 shows the 3D reciprocal modeling for a bulk GaAs crystal in *hkl* range between -3 to 3.
This algorithm is equipped with a rotation function, which is is able to rotate the bulk grain within the 3D space. It is based on the spherical coordinate system, where each bulk $\vec{q}$ is assigned by radial distance (r), polar angle (θ), and azimuthal angle (ϕ). Figure 5.10 visualizes the spherical and the Cartesian representation for a vector (the red arrow). The rotation is performed in the spherical system after transformation of the bulk grain considering the required rotation polar ($\delta\theta$) and azimuthal ($\delta\phi$) angles. Then, the coordinates of the rotated grain are transformed back into the Cartesian system. The transformation between both systems is governed by equation 5.5.

$$\begin{bmatrix} r = \sqrt{x^2+y^2+z^2} \\ \theta = \arctan\frac{y}{x} \\ \phi = \arccos\frac{z}{\sqrt{x^2+y^2+z^2}} \end{bmatrix} \underset{\text{To spherical}}{\overset{\text{To Cartesian}}{\rightleftarrows}} \begin{bmatrix} x = r\sin(\theta+\delta\theta)\cos(\phi+\delta\phi) \\ y = r\sin(\theta+\delta\theta)\sin(\phi+\delta\phi) \\ z = r\cos(\theta+\delta\theta) \end{bmatrix} \tag{5.5}$$

Figure 5.11 displays the simulated reciprocal map for two GaAs crystals having the Miller indices ranging between -3 to 3. The blue lines represent the rotated crystal with $\delta\theta = 3.9°$ and $\delta\phi = 6°$ related to the bulk one (i.e. represented by the red lines). This function will be used later to generate datasets in order to validate the clustering algorithm. The pseudo-code of the grain simulator is displayed in algorithm 3

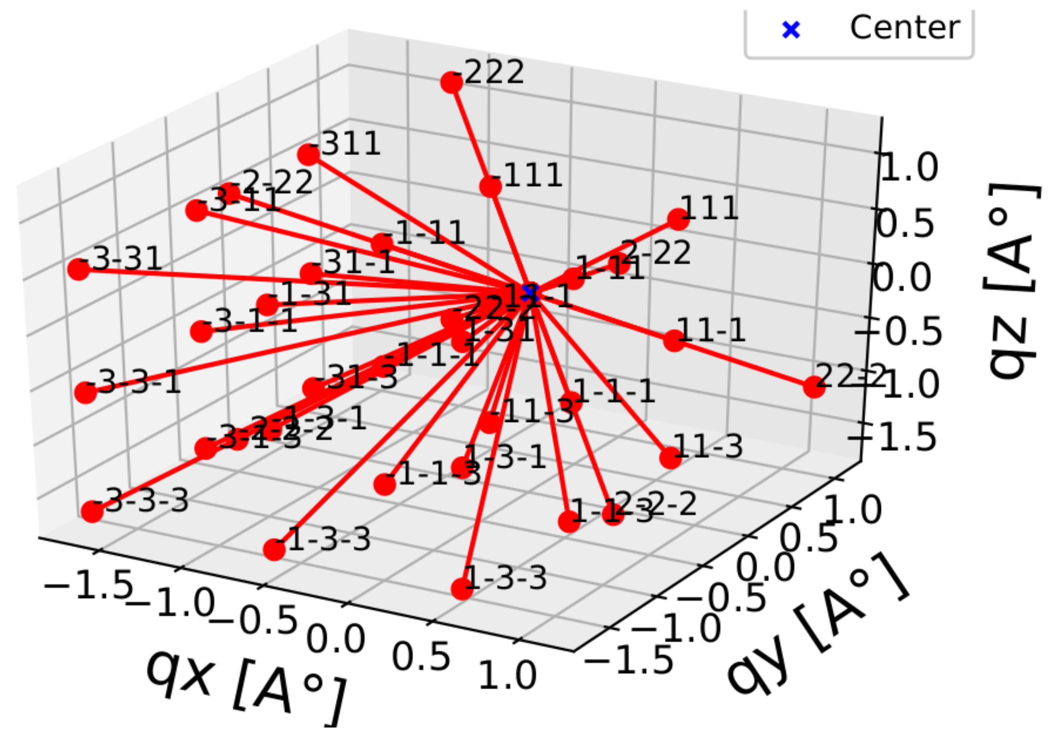


FIGURE 5.9: The simulated reciprocal map for a GaAs crystal with the Miller indices range of −3 to 3.

5.2.4.3 AT-map calculation

After simulation of the reference grain, a prior data preparation step should be performed, where all indexed reflections for an EDLD dataset are represented by an *AT-map*. It is a 3D angular space, where each $\vec{q}$ vector corresponding to an individual reflection is mapped as a function of a 3D angle, representing its orientation with respect to the predefined reference. It is essential to mention that there are two different datasets, namely the simulated and the experimental ones. For the following section, the prefix Ref is used to indicate the simulated set, while Exp is used for real data. The AT-map can be calculated as follows:

- Pairing each Exp reflection with its Ref indexing-twine one.
- Calculating the angle (ϵ) between the $\vec{q}_{Exp}$ and the $\vec{q}_{Ref}$ for each pair.
- Analysing the three components α, β, and γ of the ϵ representing the projection of ϵ on $x-y$, $y-z$, and $z-x$ planes.

The output from the *AT-map* is expected to have many groups where data-points of the same orientation to the reference grain are assigned to the same group. The number of groups represents the number of probed grains while the centroids are the rotations angles to the reference grain. By this representation, the problem is converted to be an unknown k-clustering problem, and hence, ML can be applied for its solution.

5.2.4.4 The model

As the number of cluster within the data is unknown, the traditional k-mean classifier fails to provide a solution for such datasets. Instead, it is suggested to utilize a

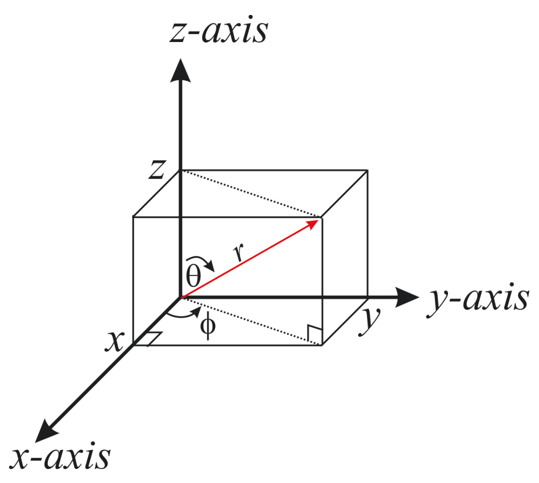


FIGURE 5.10: The spherical and Cartesian representation.

k-mean classifier which is able to treat the unknown k as a learning parameter. This part investigates two different classifiers, namely HC, and K-mean with the Elbow method. It will be shown that a combination between HC and the Elbow method might give a better solution. To achieve this, a data set of three grains having different orientation with respect to the GaAs reference grain are simulated using the explained grain-simulation algorithm with angles shown in table 5.1. Figure 5.12 visualizes the 3D mapping for all grains. The red dots represent the reference grain while black, green, and blue ones are for Grain1, Grain2, and Grain3 respectively. The *AT-map* of this dataset, shown in figure 5.13, has been fed to both mentioned classifiers and the results are shown below.

TABLE 5.1: The angles of the simulated grains to the GaAs reference grain.

	Polar angle [°]	**Azimuthal [°]**
Grain 1	5	1
Grain 2	6	18
Grain 3	3.9	6

- ***The hierarchical clustering:*** With a threshold of 3° per cluster, the HC votes for $k = 3$ to be the optimal clusters number. The dendogram of the decision tree is shown in figure 5.6
- ***K-mean with the Elbow method:*** With a k range of 0 to 9 clusters, $k = 3$ is chosen to be the Elbow point. This is in agreement with the result of the HC algorithm. Figure 5.15 displays the SEE for different k within the predefined range.

To improve the performance and the accuracy, it is suggested to use the combination of the two classifiers. The HC proposes the initial k_{init} value, which is validated by the Elbow method, whereby, the initial k_{init} is incrementally stepped by one, giving k^+_{init}. Subsequently, the traditional k-mean is performed with both k_{init} and k^+_{init}, calculating the sum of square errors SEE_{init} and SEE^+_{init} respectively. The convergence between the two SSE values is then tested to find out if it fulfills the particular threshold condition. If not, the k_{init} is updated to be k^+_{init} and this procedure is repeated till the new optimal k value is found. Algorithm 4 demonstrates the pseudo-code of

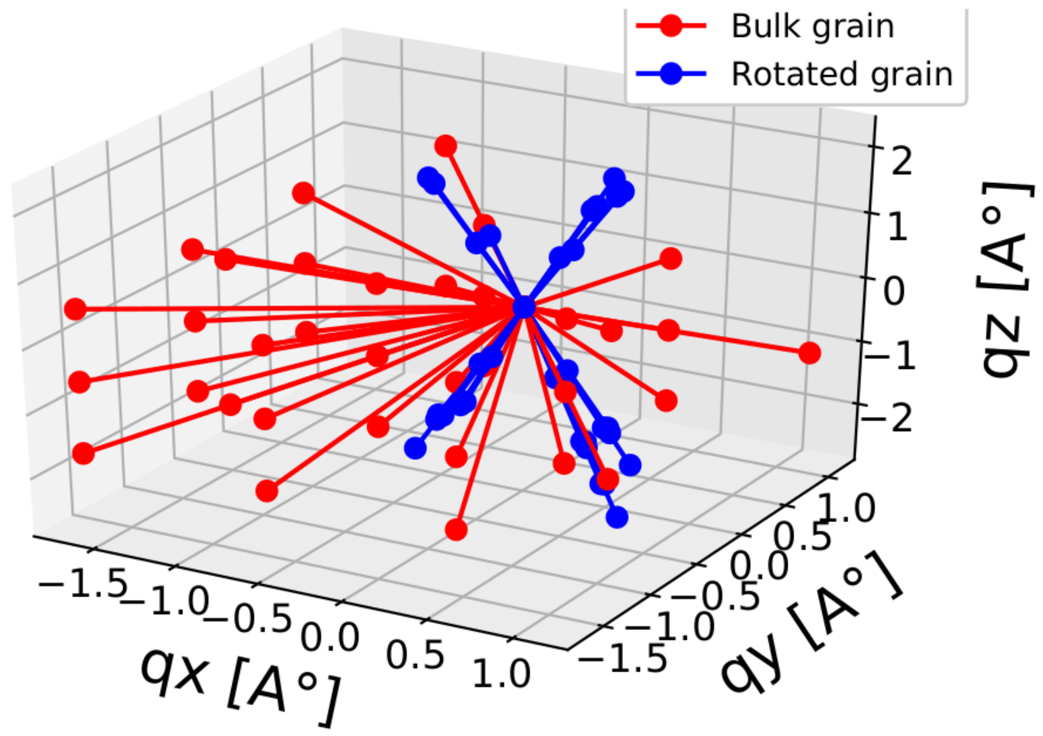


FIGURE 5.11: The simulated reciprocal map for bulk and rotated GaAs crystals with Miller indices range of -3 to 3. The red lines represent the bulk $\vec{q}$ while the blue lines are for the rotated $\vec{q}$. The rotation polar ($\delta\theta$) and azimuthal ($\delta\phi$)are 3.9° and 6°, respectively.

the model used. Implementation of this combination provides better accuracy as it makes use the advantages of two different classifiers. Moreover, this procedure is efficient for datasets that contain a high number of clusters, as it reduces the examined range of k value.

Algorithm 3 Grain simulation algorithm based on the spherical representation.

```
 1: procedure REFERENCE GRAIN SIMULATION
 2:     function DEFINE THE PARAMETERS
 3:         LC ← The lattice constant
 4:         SF ← The structure factor
 5:     function GENERATE POSSIBLE HKL GROUPS( HRange,KRange,LRange)
 6:         for h in range(-HRange,HRange)  do
 7:             for k in range(-KRange,KRange)  do
 8:                 for l in range(-LRange,LRange)  do
 9:                     CheckTheStructureFactor()
10:                     if True then
11:                         CaculateQxQyQz()
12:                     SaveAsReference()
13:         return RefArray[] ← The bulk datapoints
14:     function GENERATE THE RECIPROCAL SPACE MAP
15:     function SAVE AS DATA POINTS
16: procedure ROTATED GRAINS SIMULATION
17:     repeat
18:     for i = 0 : sizeof(RefArray[]) do
19:         ConvertToSpherical(RefArray[i])
20:         RotateToRef(RefArray[i], $\theta$, $\phi$)
21:         ConvertToCartesian(RefArray[i])
22:         SaveAsRotatedGrain()
23:     until number of required grains is reached
24:     return RotatedGrainsArray[][]
25: function VISUALIZE THE DATA
```

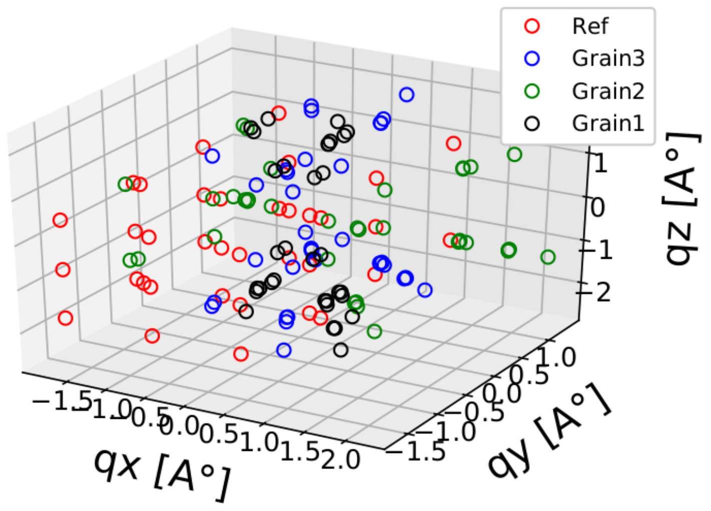


FIGURE 5.12: The reciprocal representation of all grains.

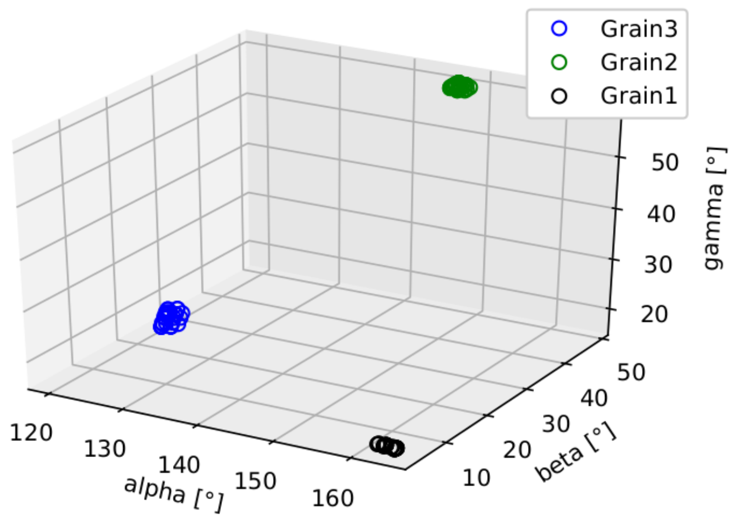


FIGURE 5.13: The *AT-map* of the simulated dataset.

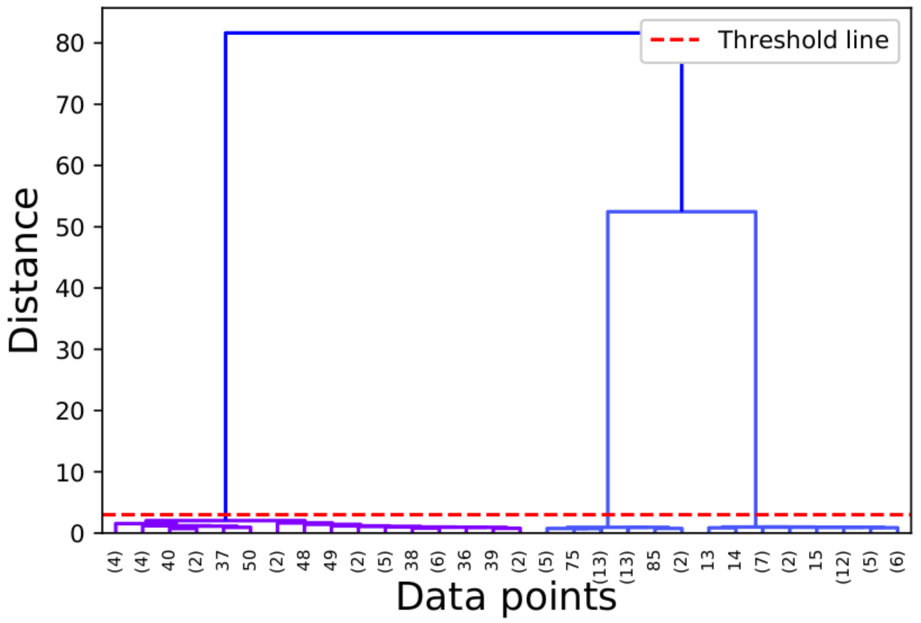


FIGURE 5.14: The dendogram of the HC algorithm.

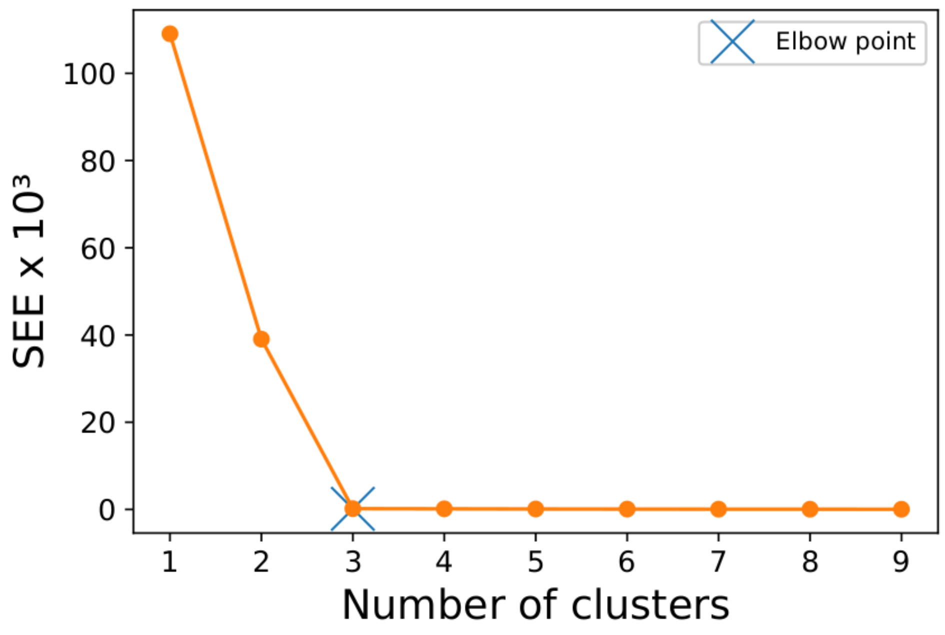


FIGURE 5.15: The diagram of the k-mean algorithm with k range of 1 : 9 clusters. Based on the result of the Elbow technique, the number of cluster in this dataset is 3.

Algorithm 4 Grain simulation algorithm based on the spherical representation.

```
 1: DataArray[] ← The AT-map data
 2: function RUN THE HC ALGORITHM(DataArray[])
 3:     return $k_{init}$ ← The initial k value
 4: let IsElbow $= false$
 5: while IsElbow $= false$ do
 6:     let $k^{+}_{init} = k_{init} + 1$
 7:     function RUN THE K-MEAN ALGORITHM($k_{init}$)
 8:         return $SEE_{init}$
 9:     function RUN THE K-MEAN ALGORITHM($k^{+}_{init}$)
10:         return $SEE^{+}_{init}$
11:     function CHECK CONVERGENCE($SEE_{init}$, $SEE^{+}_{init}$)
12:         return IsElbow
13:     if IsElbow == false then
14:         $k_{init} = k^{+}_{init}$
15:     if IsElbow == true then
16:         return IsElbowThe final k value
```

5.2.4.5 Experimental datasets and discussion

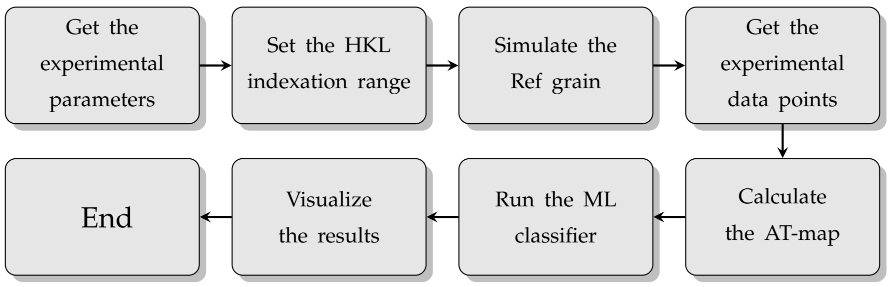

Get the experimental parameters
Set the HKL indexation range
Simulate the Ref grain
Get the experimental data points
Calculate the AT-map
Run the ML classifier
Visualize the results
End

FIGURE 5.16: Analysis procedure pipeline.

After the verification of the suggested model by the simulated dataset, it has been tested on the data collected by the previously conducted experiments. The procedure pipeline is shown in figure 5.16. First, the experimental setup parameters (i.e. the detector position to the primary and to the sample,the beam energy range, the probed material structure, the detector and pixels size) are sent to the model and are used for further calculations. Starting from equation **??**, the HKL range is estimated:

$$-\left[\frac{2.a.\sin(\theta_{max})E_{max}}{hc}\right]^2 < h^2 + k^2 + l^2 < \left[\frac{2.a.\sin(\theta_{max})E_{max}}{hc}\right]^2 \tag{5.6}$$

where E_{max} is the maximum photon energy provided by the radiation source and $\sin(\theta_{max})$ is calculated according to equation 5.7:

$$\sin(\theta_{max}) = \frac{XY}{\sqrt{XY^2 + STD^2}} \tag{5.7}$$

where STD is the travailing distance between the sample and the detector, XY is the absolute distance between the reflection and the primary beam within the detector plane calculated by equation 5.8:

$$XY = \sqrt{\left[(x_{dec} - x_{beam}) * Pix_{x-size}\right]^2 + \left[(y_{dec} - y_{beam}) * Pix_{y-size}\right]^2} \tag{5.8}$$

where x_{dec} and y_{dec} are the horizontal and vertical detector size in pixel, respectively. The primary beam pixel position is given by x_{beam} and y_{beam}, while Pix_{x-size} and Pix_{y-size} are the square pixel sizes. The reference bootstrap data are simulated as explained in the previous part. The reflections of the recored Laue pattern are listed and sent for the *AT-map* representation. Each reflection is defied with its Miller indices (i.e. h, k, and l) and scattering vector components (i.e. q_x, q_y, and q_z). Finally, the *AT-map* is determined and fed to the classifier. This procedure has been performed for two different experimental datasets:

1. Single crystal dataset: A single crystalline GaAs sample was mounted at the beamline of the BESSY II storage ring in Berlin and the Laue pattern was recorded in transmission geometry by a pnCCD [105]. Figure 5.17 visualizes the Laue pattern recorded in a single exposure of the $STD = 41mm$. The expected range of the Miller indices is between -20 and 20. For simplicity reasons, a decision was taken to reduce this range to be $-10 : 10$ and only the indexed points which meet this range are processed. The reciprocal mapping is shown in figure 5.18, the red and blue dots are for the bulk grain and experimental data points respectively. Figure 5.19 visualizes the *AT-map* of

this dataset including the $(0, 0, 0)$ point for better visualization. The HC dendogram is shown in figure 5.20, where the dashed red line is the predefined threshold. As all the found clusters are below the threshold line, the classifier has merged all data points into one cluster giving $k = 1$. Based on the feeding of the Elbow k-mean model with the *AT-map* and $k_{init} = 1$, the Elbow point was voted to be 1. Figure 5.21 displays the SSE variation to the number of clusters during the verification procedure. Ultimately, this dataset was predicted to have only one cluster with the centroid shown in figure 5.22. This indicates that the examined Laue pattern contains reflections generated by a single crystal. This result is totally in agreement with the expectations.

2. Polycrystalline dataset: An EDLD experiment was performed using a polycrystalline Nickel wire [134] and the Laue pattern was collected by a pnCCD. Several reflections were selected and processed. Figure 5.23 shows the Laue pattern and the assigned reflections. According to the prior analysis [134], the reflections belong to nine different grains. The same procedure was applied on this dataset. Figure 5.24 shows the 3D *AT-map* fed to the classifiers. The dendogram of the HC is displayed in figure 5.25. The threshold line has 9 intersection points with the plotting, giving $k_{init} = 9$. The Elbow k-mean diagram is shown in figure 5.26, where the voted Elbow point is 9. Conclusively, nine grains have been identified in this dataset which supports the results of with the previous analysis.

5.2.4.6 Technical description and conclusion

This algorithm has been developed using Python programming language. The classifiers are based on SciPy library [135]. The 3D and 2D plots have been generated exploiting Matplotlib library [136]. To criticize the performance in real-time, a latency analysis has been performed on the three components of this software (i.e. grain simulation, *AT-map* calculator, and classification). Figure 5.28a demonstrates the elapsed time by the grain simulator to generate the reference grain with different HKL ranges. Practically, a typical EDLD is not expected to exceed the HKL range of $-20 : 20$ because of the limitations of the experimental setups (e.g. the beam energy, detector size, and detector quantum efficiency). Depending on this fact, the maximum latency required to simulate a reference grain is less than $1sec$. The elapsed time of both *AT-map* calculator, and classification procedure have been investigated and shown in figure 5.28b and 5.28c respectively. This has been done assuming a reference grain with the HKL range of $20 : 20$. Basically, the latency of the classification procedure can be neglected as it has no major effect. On the other hand, the *AT-map* calculator shows slow performance in the case of extremely crowded Laue patterns (i.e. having more than 300 reflections). This behavior is diminished with the number of processed reflections less than 300.
This part has shown an ML-based solution to identify grain-corresponding Laue reflections in a polycrystalline ED Laue pattern. It has demonstrated that data points representation is the key to any successful classification algorithm. The *AT-map* representation offers an innovative approach to prepare EDLD datasets. Furthermore, it has been proven that a combination of two classifiers (i.e. HC and the Elbow k-mean) boosts the performance and the accuracy of the clustering procedure. The latency analysis has emphasized the real-time properties of the developed algorithm.

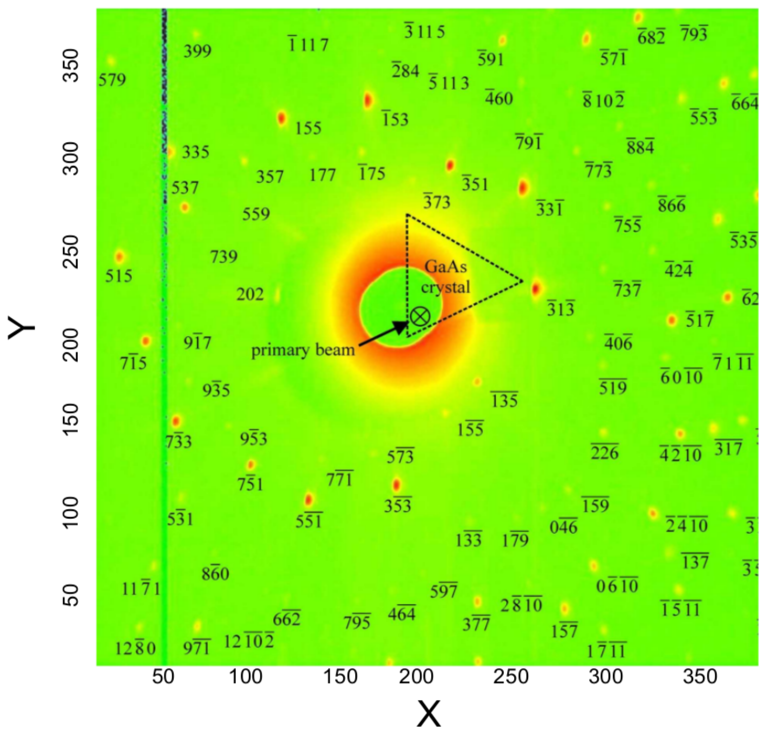


FIGURE 5.17: The indexed Laue pattern of GaAs single crystal. Only the Bragg peaks with the highest integrated intensity were assigned to the identified Laue spots in the figure.

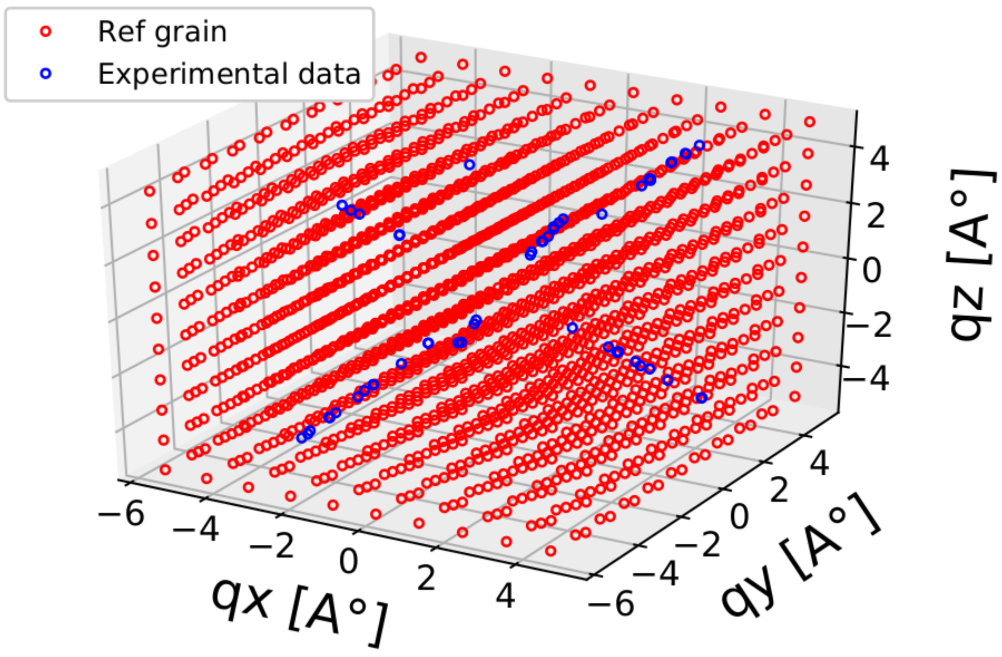


FIGURE 5.18: The 3D plotting of the reciprocal mapping for a GaAs single crystal. The red dots represent the simulated data while the blue dots are for the experimental data points.

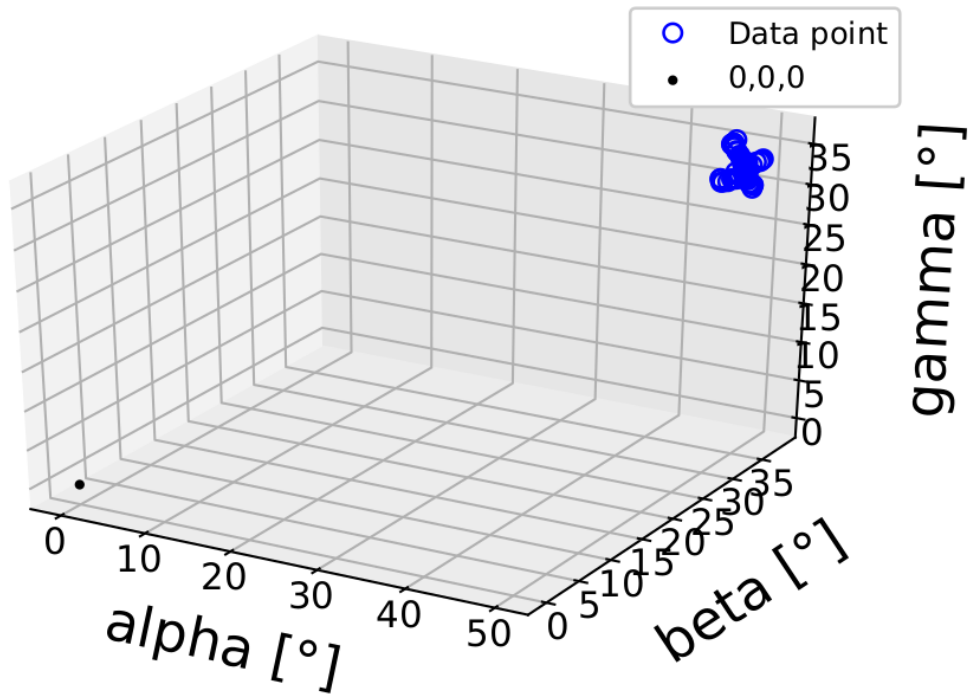


FIGURE 5.19: The *AT-map* of the experimental data points. The $(0,0,0)$ point was included for better visualization.

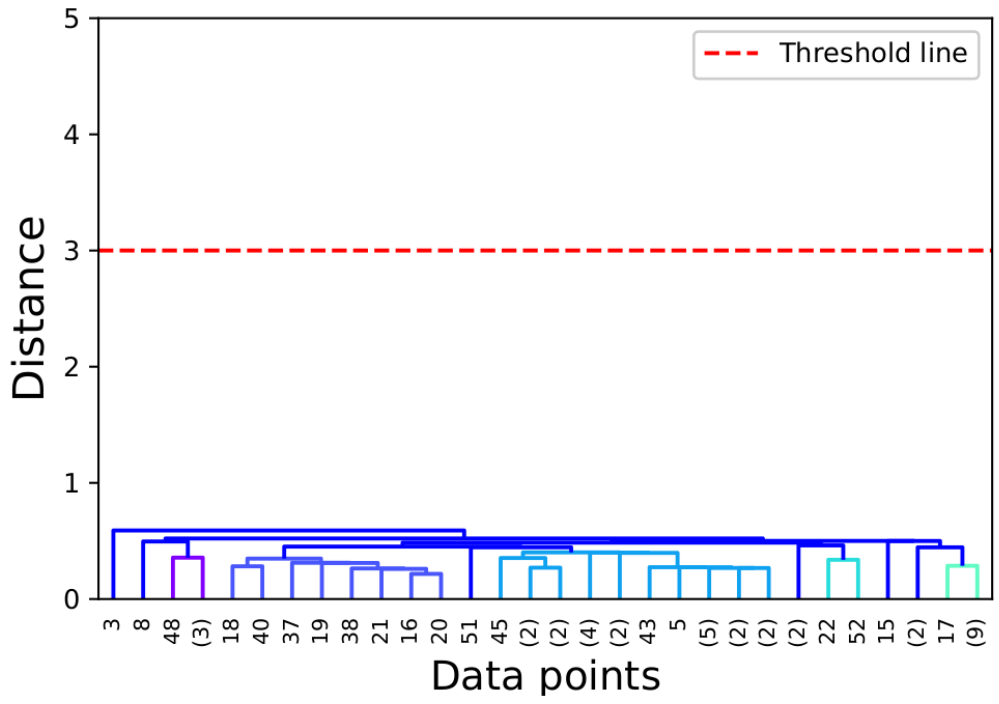


FIGURE 5.20: The dendogram of the HC for a GaAs single crystal dataset. The dashed line is the predefined threshold level. According to this threshold, all data points have been merged into a single cluster.

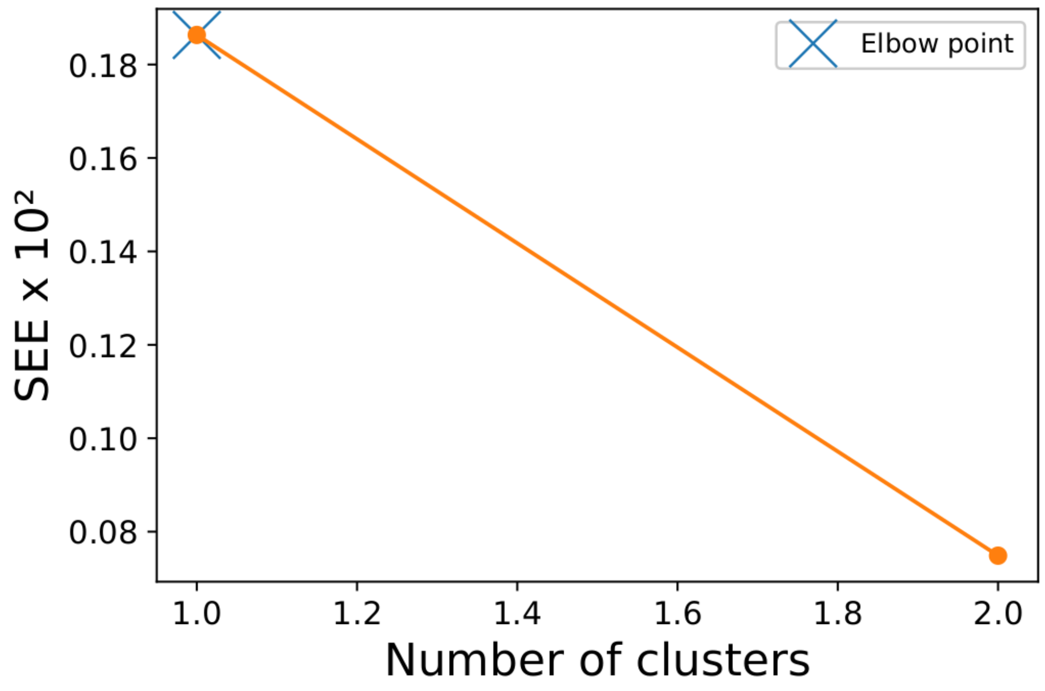


FIGURE 5.21: The diagram of the Elbow k-mean algorithm with $k_{init} = 1$. The Elbow point was voted to be 1.

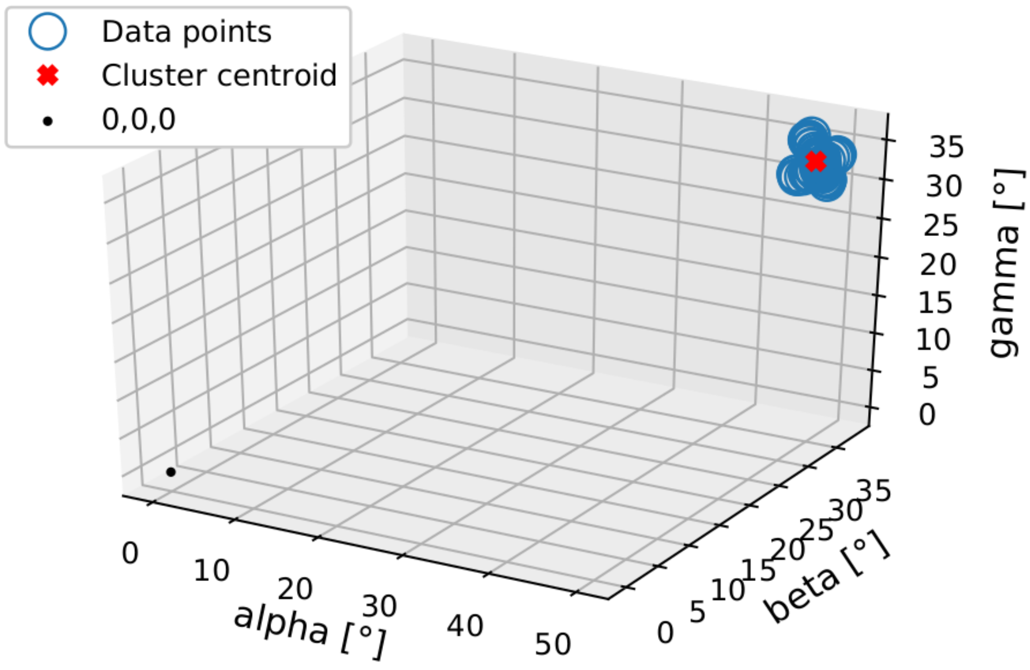


FIGURE 5.22: The clustered *AT-map* of the GaAs sigle crystal dataset. The red *x* represents the centroid of the predicted cluster.

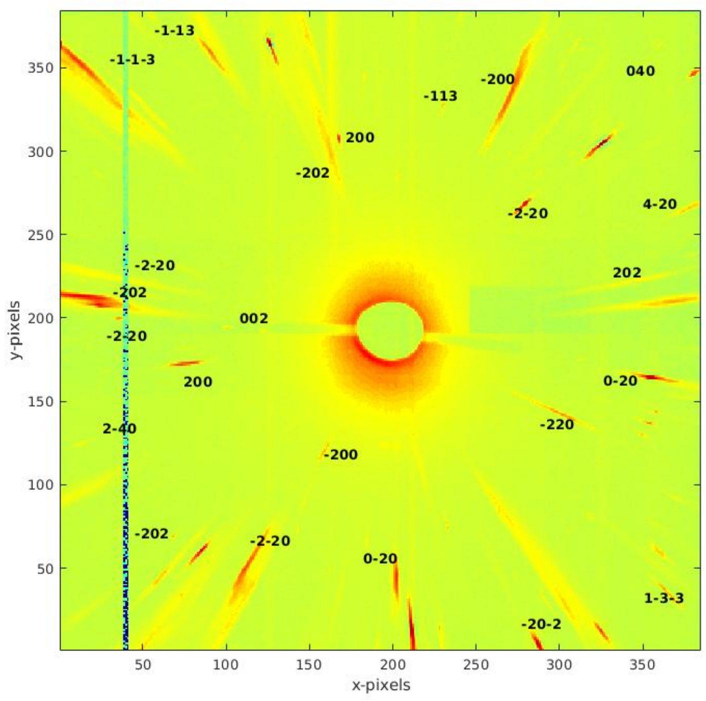


FIGURE 5.23: The indexed Laue pattern of Ni polycrystalline sample.

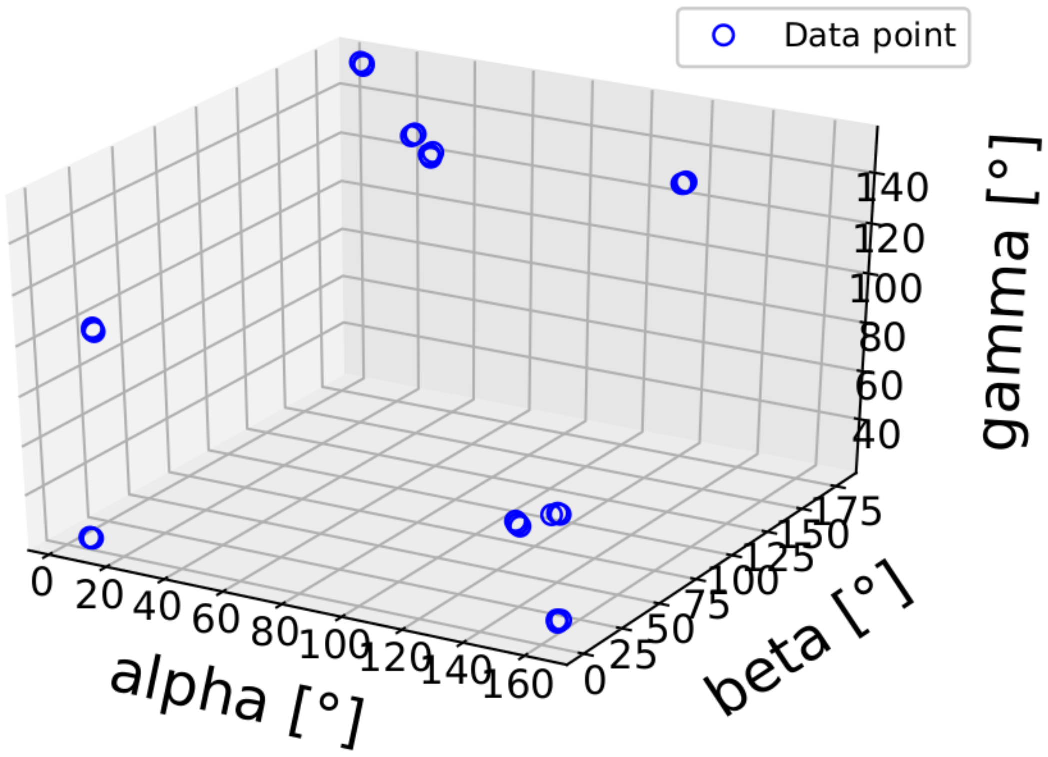


FIGURE 5.24: The *AT-map* of the experimental Ni polycrystalline sample data points.

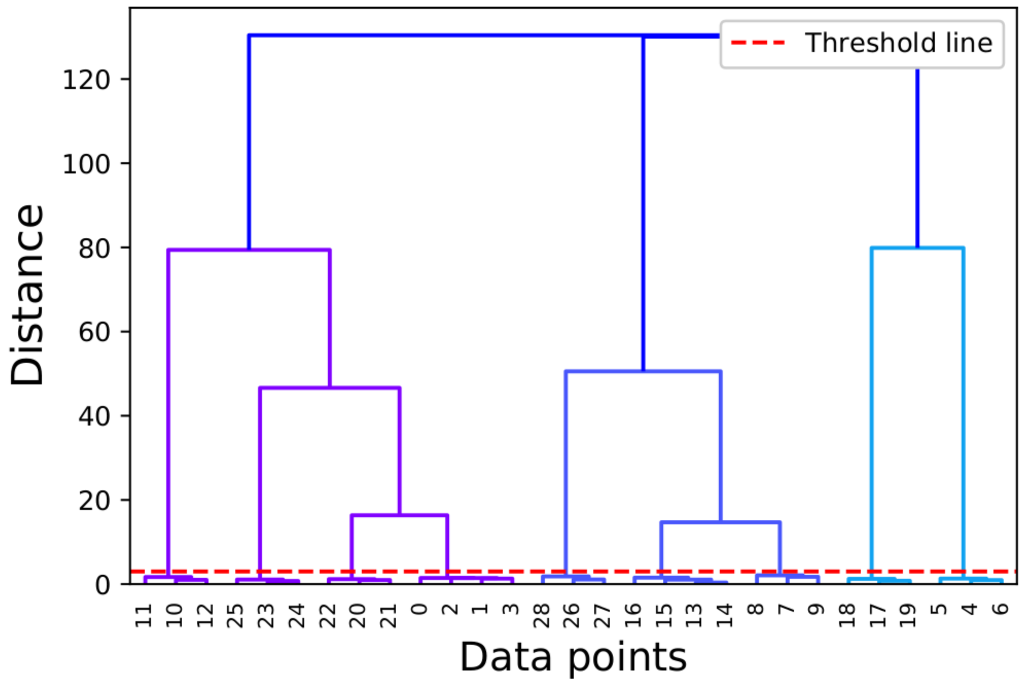


FIGURE 5.25: The dendogram of the HC for a Ni polycrystalline sample dataset. The dashed line is the predefined threshold level. According to intersection between the threshold and the plotting, this dataset was clustered into 9 ensembles.

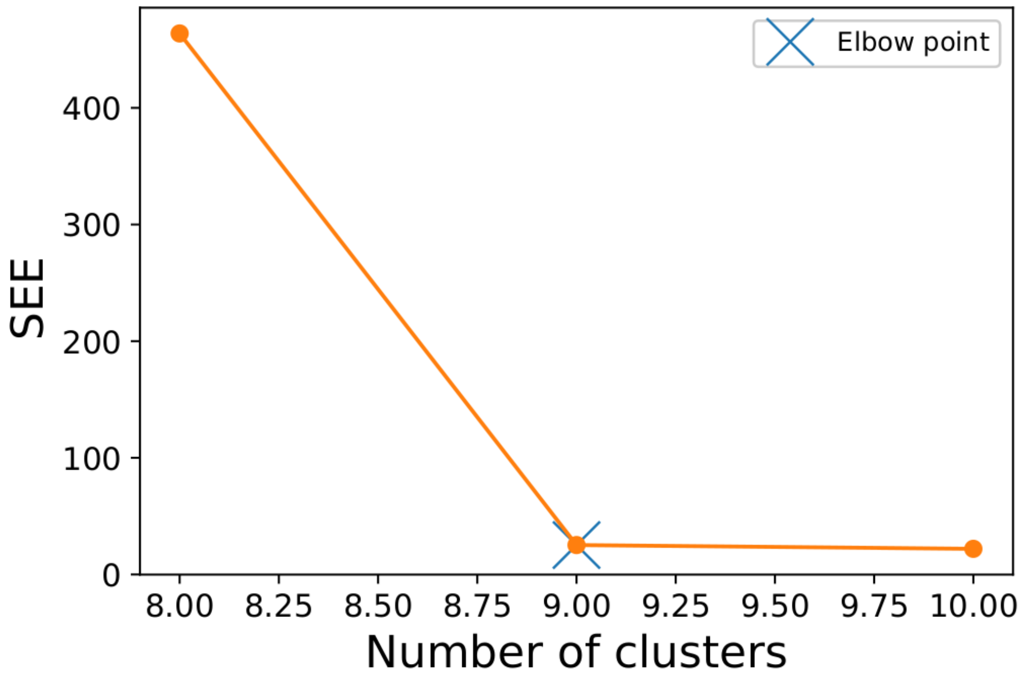


FIGURE 5.26: The diagram of the Elbow k-mean algorithm with $k_{init} = 9$. The Elbow point was voted to be 9.

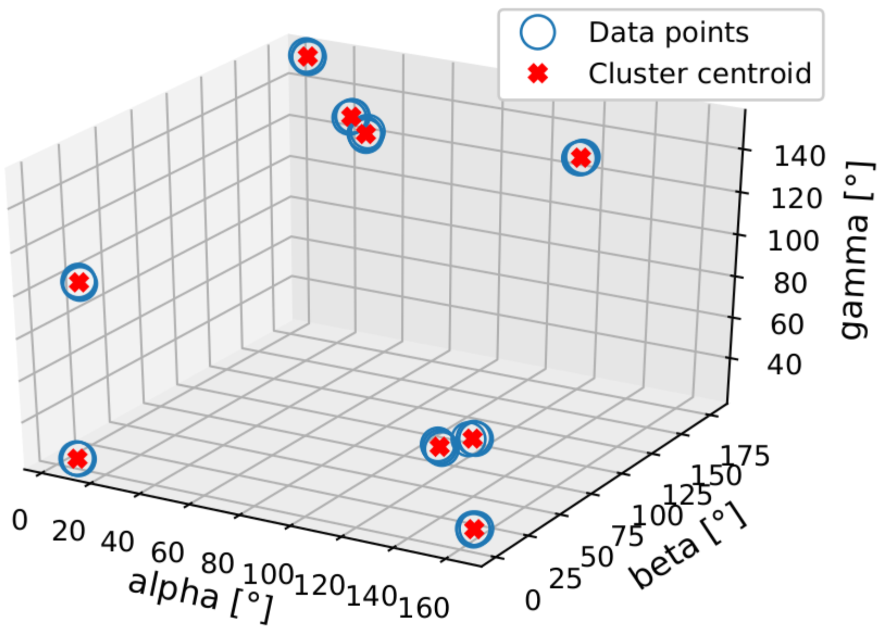
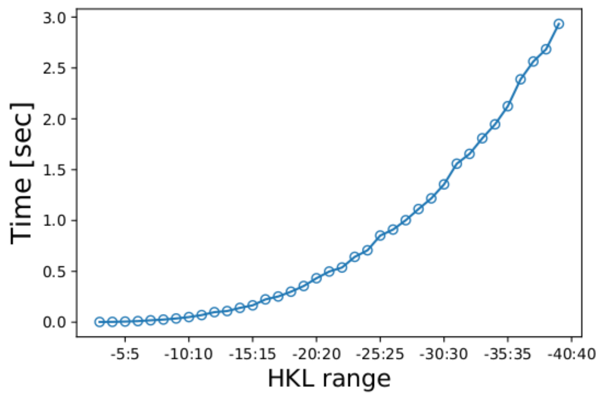
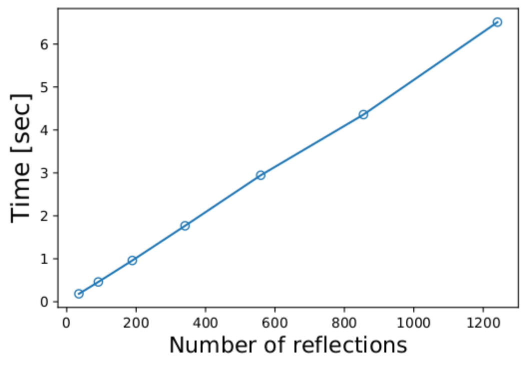


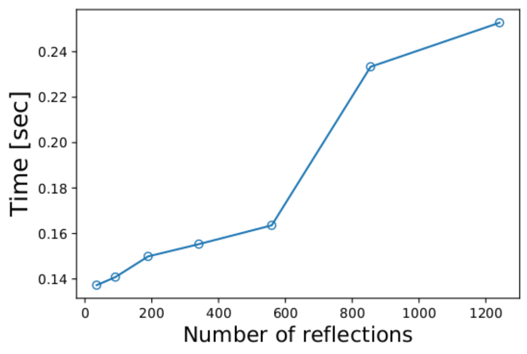
FIGURE 5.27: The clustered *AT-map* of the Ni polycrystalline sample dataset. The red *x* represents the centroids of the predicted clusters.



(A)



(B)



(C)

FIGURE 5.28: The time to process different number of reflections within a Laue pattern. (A) is the elapsed time to simulate the reference grain vs the *HKL* ranges. (B) is the latency of the *At-map* calculator, while (C) is for the classification procedure.

5.3 Deep Learning (DL)

Many scientific studies have adopted DL techniques to find solution for such problems as strain imaging from nano-crystals [137], crystal structure classification [138], Deep Learning Methods for Visual Fault Diagnostics of dental X-ray systems [139] and imaging nanoscale lattice vibrations [140]. The idea behind the implementation of DL methods in data analysis is s to make use of their considerable advances in computation and management of very large datasets [141]. Furthermore, DL methods have recently been shown to exceed human performance in performing visual tasks such as object recognition and image classification [142]. This section is to briefly describe the state of the art in DL methods and to investigate their applicability for automatic event classification collected by a 2D energy-dispersive detector (e.g. pnCCD). This implementation might improve both quality and performance of the event reconstruction procedure for such detectors. To fulfill the above stated goal, first, the main DL concepts are explained in subsection 5.3.1. Next,the focus is shifted to the Convolutional Neural Networks (CNN), which are particularly relevant for this work (subsection 5.3.2). In subsection 5.3.3, some current trends in DL are explained. Finally, subsection 5.3.4 presents its application of DL in the crystallography community.

5.3.1 Deep learning basics

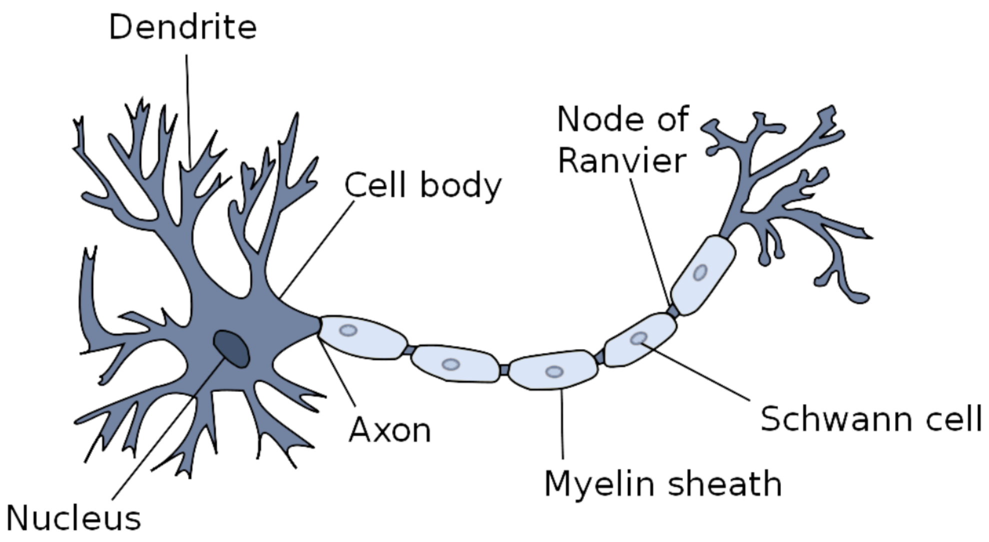
5.3.1.1 4.3.2.1 The neuron model

The neuron model is inspired by the biological neurons that shape the networks of brains of living creatures. Therefore, it is important to introduce a basic summery of these natural systems. A neuron is a fundamental element of the neural network. It is considered to be an information-processing cell, which transmits signals between input and output destinations [143]. As shown in figure 5.29a, a neuron consists of three constituents: dendrites, cell body (i.e. Soma), and axon. The dendrites are the information-receivers of a neuron. They have a tree-like shape where each branch is a connection line linked with the surrounding neurons by the weighting synapses. The cell body (soma) is the core where all received signals are accumulated and stored for further transferring. Once the accumulated signals fulfill a certain condition (i.e. exceeding the threshold value), an electrical impulse is fired by means of the axon and transferred to the output destinations based on previously learned patterns. The axon terminals, located at the end of the axon's branches, connect spread signals across the synapses of other neurons. A set of linked neurons represent a neural network (NN) or neurons system. The neural capacity of a NN is proportional to the capabilities of the system. For instance, a human being, who has cognitive capabilities (e.g. speaking, remembering, abstracting, mechanizing) needs 10^{11} neurons, while an ant, having limited capabilities, requires 10^4 neurons [143, 144]. From a computational perspective, a NN might be modeled by the mathematical form given in equation 5.9.

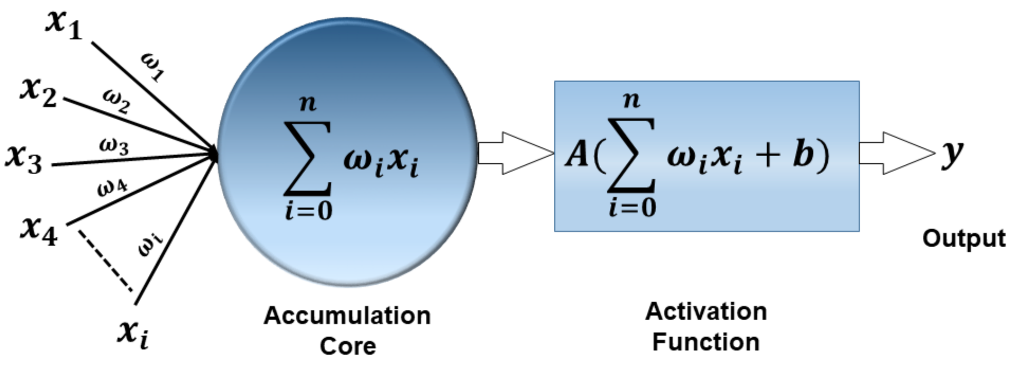
$$y = A(\sum_i \omega_i \vec{x}_i + b) \tag{5.9}$$

where y is the scalar output transmitted by the axon terminals, $\vec{x}_i$ is the vectorial input of the neuron ith, ω_i is the synaptic weight of the neuron ith, and b is the cell body bias. $A()$ is the firing function, which is often termed as *"activation function"*. The sum of all input signals refers to the accumulated signal stored in the soma.

Based on the model of natural neuron, the first perceptrons (i.e. artificial neurons) were formulated by Frank Rosenblatt [145].



(A)



(B)

FIGURE 5.29: The neuron models, (A) is biological neuron structure [143], while (B) is the imitative artificial perceptron.

5.3.1.2 Artificial Neuron Network (ANN)

An ANN is a connectionist system where several perceptrons are homogeneously operating together. The main objective of such systems is to perform the cognitive functions of the human brain. Each individual neuron is called a *"node"* and has its own bias (b). It is responsible for extracting, learning, or estimating a specific feature from the weighted vectorial input. The number of implemented perceptrons and their interconnected structure is called *"the architecture of the neural network"* [146].

a) ANN Layers: The neurons of an ANN are grouped in subsets, so-called *"layers"*. There are three types of layers, namely input, output, and working layers. As their names suggest, the input and output layers are the communication channels between the ANN and the outer components. The working layers are often labeled as *"hidden layers"* as they are invisible to the external systems. They are the layers in between the input layers and the output layers, which are responsible for extracting a different set of high-level features. ANNs are categorized according to the number of their hidden layers or signal propagation.The classification based on the amount of hidden layers contains two types: single-layer, and multi-layer ANNs. It is essential to mention that stacking different neural network layers in a multi-layer ANN is termed as *"Deep Learning (DL)"*. On the other side, a single layer ANN is considered as a *"shallow learning"* architecture. The signal-propagation-based definition includes various types of ANNs architectures, such as feed-forward, Radial Basis Function (RBF), Recurrent Neural Network(RNN), and Convolutional Neural Network (CNN). This work focuses only on the CNN architecture, more details about the other kinds can be found in [146–149].

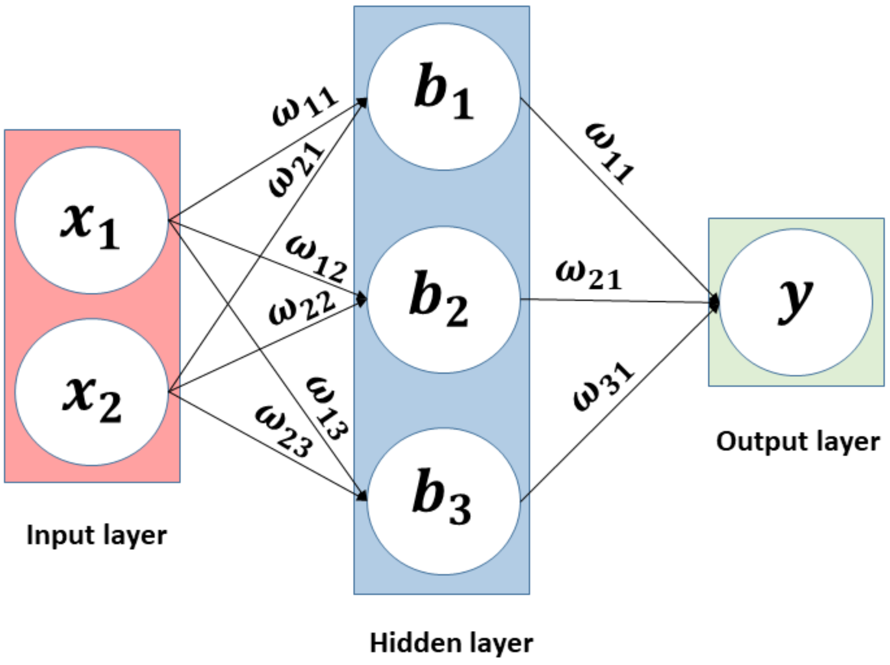


FIGURE 5.30: A simple feed-forward neural network with one hidden layer.

b) The notation for the weights and biases: The weight and the bias indicate the influence of the perceptrons on a NN: the higher the weight is, the higher is the domination. The bias is an extra input to neurons, representing the offset of the neuron. The commonly used notations for the weights and the bias of an ANN are ($\omega_{mn}^{[l]}$) and ($b_n^{[l]}$) respectively, in which l is the layer number, while m and n refer to the number of the launch and destination nodes in the lth layer respectively. For an example, the output of the ANN shown in figure 5.30 can be given as:

$$y = A(W^{[1]}\vec{x} + B^{[2]})W^{[2]} \tag{5.10}$$

where $A()$ is the activation function,

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, W^{[1]} = \begin{bmatrix} \omega_{11}^{[1]} & \omega_{21}^{[1]} \\ \omega_{12}^{[1]} & \omega_{22}^{[1]} \\ \omega_{13}^{[1]} & \omega_{23}^{[1]} \end{bmatrix}, B^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{bmatrix}, W^{[2]} = \begin{bmatrix} \omega_{11}^{[2]} \\ \omega_{21}^{[2]} \\ \omega_{31}^{[2]} \end{bmatrix}$$

During the training phase, an ANN is trained to find the values of all weights and biases involved in the architecture. This helps the NN to find a function which maps input to the associated output label.

c) Activation function: The artificial neuron activation is the switching-status gate of a neuron. It takes an vectorial input and performs point-wise operation, which fires a binary output (i.e. [0,1] or [-1,1]) [139]. This allows to overcome the non-linearity between a response variable and its input variables. The capacity of any ANN principally depends on the result of its activation function. In the part below, some popular and commonly used activation functions are shown.

- **Sigmoid or logistic:** It is standard logistic non-linear function or quasi step function. It bonds the resultant output values between 0 and 1, normalizing the output of a neuron. Figure 5.31a displays the curve diagram of the sigmoid function. The most serious disadvantage of this activation technique lies in the vanishing gradients in case of strongly-negative input. This causes a neural network to get stuck at the training time. Sigmoid activation function is governed by the following mathematical form:

$$y = S(x) = \frac{1}{1+e^{-x}} \tag{5.11}$$

- **Hyperbolic tangent (tanh):** It is a sigmoidal function (i.e. S-shaped), squashing outputs values in the range between -1 and 1. It is a zero-centered function that makes it easier to deal with negative, neutral, and positive input values. Basically, it also faces the vanishing gradients problem at the edges. Figure 5.31b shows the curve of the *tanh* activation function, while the following equation is the mathematical form:

$$y = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2S(2x) - 1 \tag{5.12}$$

- **Rectified Linear Unit (ReLU):** ReLU is the most commonly used activation function in neural networks. It is a linear function for the positive input and zero for the negative one. In other words, the firing output for the negative input is clipped to zero, while it remains the same for the positive input. It has a very high computationally efficiency, allowing the fast convergence during the training procedure. Figure 5.31c is the curve of the *ReLU* activation function. It has the following equation:

$$y = Re(x) = max(0, x) \tag{5.13}$$

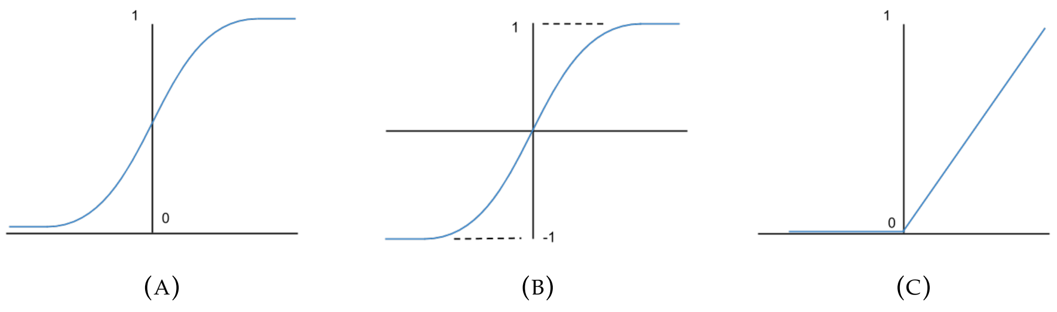


FIGURE 5.31: The typical activation functions curves. The left diagram (A) is sigmoid. The middle (B) and the right (C) onces are for tahn and ReLU, respectively.

d) Loss function: A loss function or cost function (ρ) is an evaluation function to measure the deviation of the learner performance from the real value. At its core, this loss function simply measures the absolute difference between the prediction and the actual value. It might have the mathmatical notation: $|y_{predicted} - y_{actual}|$. A loss function learns to be minimized, giving an indicator of *"how good the ANN is"*. Often, the loss function is distinguished into two major types depending on the learning task: regression losses and classification losses.

e) Feedforward propagation: In forward propagation, an ANN has an open topology, where the signal is fired in one direction; from the input layer to the output layer [150]. There is no possibility for the neural network to correct itself after predicting the output. This results in high loss in case of non-linear correlated data and learning-limitation, such as a failure to learn in case of sequential or time-depended data. Therefore, the multi-layer perceptron (MLP) is widely used, allowing to learn non-linearly separable patterns.

f) Back propagation: It is an innovative technique to minimize a loss by using a closed-topology ANN. The signal is fired in two opposite directions: 1) calculating the derivatives (the so-called *"gradients"*) of the losses to the weights of the last layer. 2) sending the feedback in the opposite direction, gradually updating the previous layers. Basically, back propagation uses the chain rule of differential calculus, where it iteratively processes batches of data through the NN then updating the weights. This can effectively help to steer the learner, in a descending manner, towards the losses local minima. This algorithm is known as Gradient Descent (GD).

5.3.2 Convolutional neural network (CNN)

A CNN or ConvNet belongs to the DL architecture, where the perceptrons have learnable weights and biases. It is very efficient in signal processing and image-related applications (i.e. image classification, object recognition, and image generation). Conceptually, it is governed by computer vision and neural networks [151]. The first CNN was implemented in 1998 when the *"LenNet-5"* network architecture was published by Y. LeCun [152]. Figure 5.32 shows the original model architecture of LenNet-5 network. Thanks to the new technologies in the computing field, many CNN architectures have been developed that efficiently process huge volume inputs , such as GoogLeNet [153], and VGG [154]. A CNN can have the following layers:

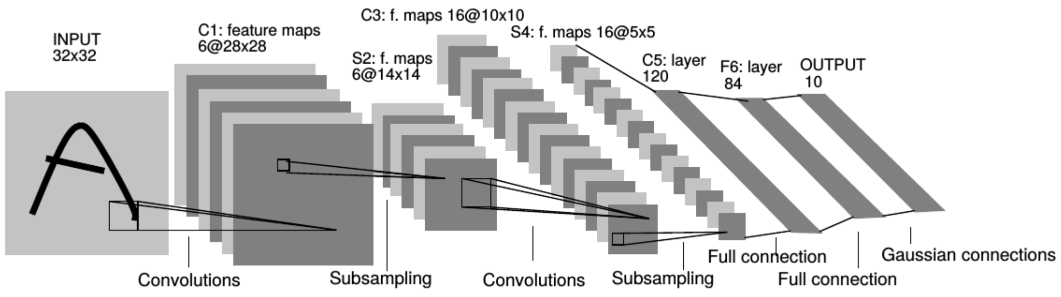


FIGURE 5.32: The model architecture of LenNet-5 network [152].

5.3.2.1 Convolution layer

A convolution layer is specialized to extract patterns from the input images and make sense of them. It can detect the relation between a pixel and its neighboring pixel within a predefined area, preserving the spatial properties between pixels. This can be done by applying convoluting filter mask (i.e filter kernel, feature detector) on the original input image. A filter mask is a coefficients matrix, epitomizing a specific feature, such as image edges, object boundaries, pixel saturation, texture objects, etc. During the convolution process, a predefined filter proceeds on local regions in the input (e.g. defined by the size of the filter), computing a dot product between the filter coefficients and the proceeded area. Consequently, a features map is generated as an output of the convolutional layer, containing important features of the original image. Figure 5.33 shows a simple example of a convolution layer. A 2×2 filter is applied on a 4×4 input map. Each convoluted local input area results in an element within the convoluted features map. By each step the filter proceeds to another local area with a predefined shifting step. To perform a successful convolution, three hyper-parameters should be defined, namely:

- The kernel size, which refers to the window size surrounding the local input area being convoluted.
- Stride, which defines the step by which the kernel jumps each iteration.
- Detection features, which are the mathematical coefficients of the applied filter.

Figure 5.34 displays the coefficients matrices of the commonly used filters. The first two matrices in the left-hand part are the typical vertical and horizontal edge detectors. Figure 5.34c and 5.34d are the so-called *"Sobel"* and *"Scharr"* filters respectively. Figure 5.35 shows the effect of applying the Sobel filter to calculate the X and Y partial derivatives of the original image. Figure 5.35a is the original image [155], while figure 5.35b is the features map. Moreover, the weights of the filter can be treated as learning-parameters instead of learning from each pixel-wise weights. This can reduce the number of learning-parameters during the training procedure enhancing the computing performance.

5.3.2.2 Pooling (down-sampling) layer

A pooling layer is another building block of a CNN. It operates on each feature map independently. Its function is to gradually minimize the size of the features map, engaging no extra parameters. This can reduce the amount of training-parameters and computation resources in the network. It assembles several elements from a

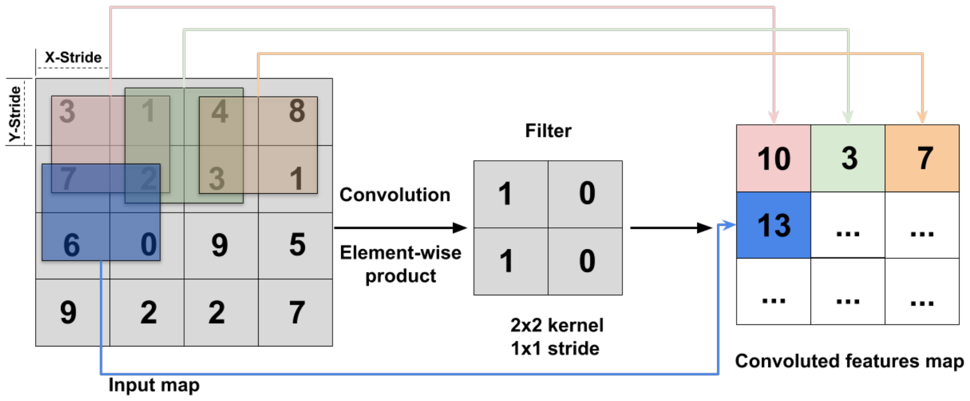


FIGURE 5.33: A convolution layer scheme. The original features map has a size of (4x4). The applied filter layer has kernel size of 1x1 and stride of 2x2. Each element within the output convoluted map is the element-wise product of the filter and the corresponding features map elements.

1	1	1
0	0	0
-1	-1	-1

(A)

1	0	-1
1	0	-1
1	0	-1

(B)

1	0	-1
2	0	-2
1	0	-1

(C)

3	0	-3
10	0	-10
3	0	-3

(D)

FIGURE 5.34: The common convolution filters. (A) and (B) are the vertical and horizontal edge filters respectively. (C) is the Sobel filter while (B) is the Scharr filter.

features map into an individual output element by applying a pooling function. The hyper-parameters of a pooling layer are the following:

- The pooling kernel size
- Stride
- Pooling function, which is the mathematical operator applied on the chosen elements.

The common pooling function is named *" Max Pool"* gives an output to the maximum element among the kernel elements. *" Average Pool"* is another kind of the pooling function. The resultant output is the average value of the kernel elements. Figure 5.36 displays an unpretentious example of the Max pooling function, with the 2x2 kernel and 2x2 stride, applied on the 4x4 features map. The pooled features map is then minimized to be 2x2.

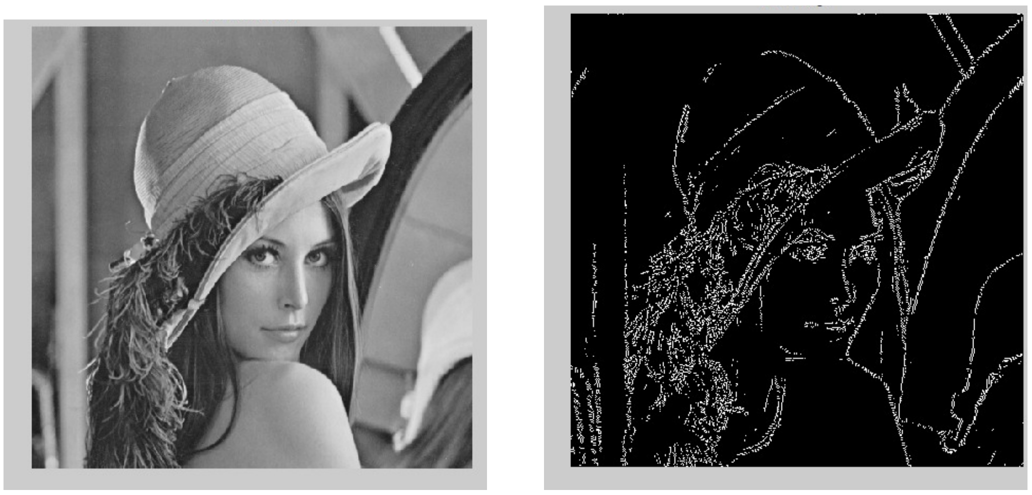


FIGURE 5.35: The Sobel edge detector. The left image is the original image, while the right one is the output after applying the Sobel filter, Source: [155].

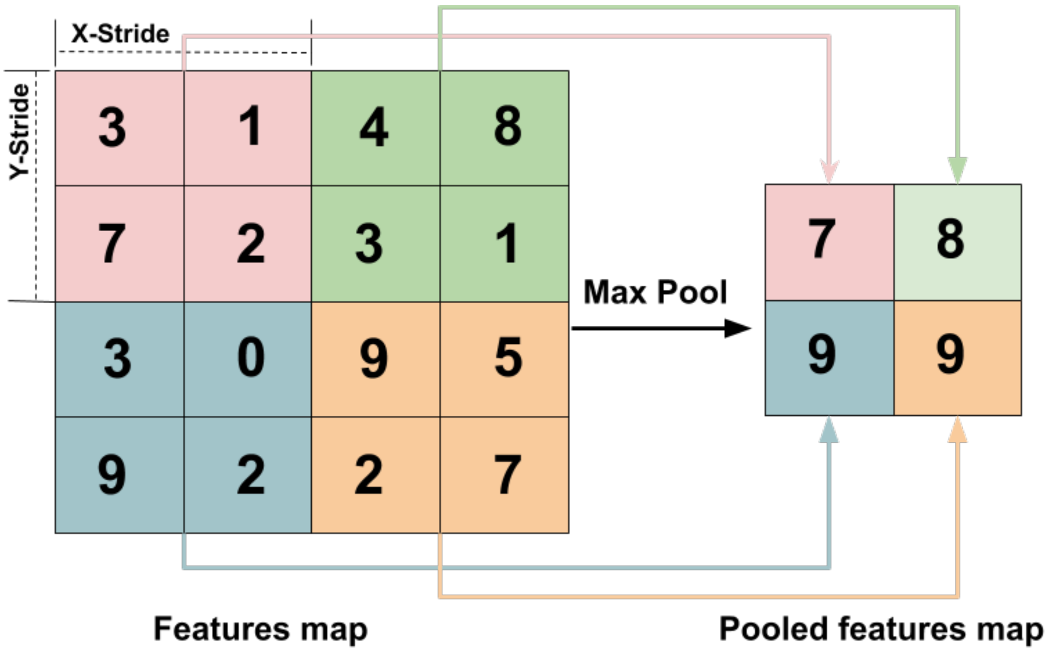


FIGURE 5.36: The Max Pool layer scheme. The original features map has a size of (4x4). The applied pool layer has a Max pooling function with kernel size of $2x2$ and stride of $2x2$. The output pooled map is shrunk to have a size of $2x2$.

5.3.2.3 Flattening layers

The main task of flattening or vectorization layers is to transform an N-D array into 1-D array, avoiding the implicit iteration over an N-D array. This can save computation resources and reduce the training latency. Basically, flattening layers have no considerable effect on the performance of the trained module. Figure 5.37 visualizes a scheme of a flatten layer applied to an 2D ($3x3$) array. The output is an 1D array having 9 elements that is passed through the ANN for further processing.

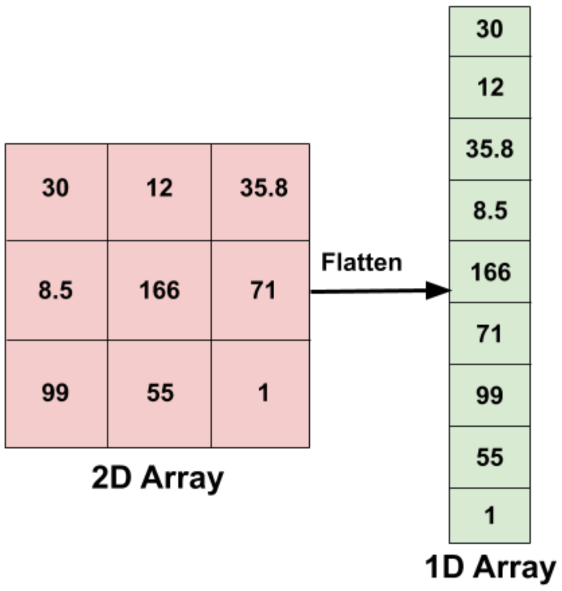


FIGURE 5.37: The flatten layer scheme.

5.3.2.4 Fully Connected (FC) layers

Fully connected layers are substantial components of any CNN. The main purpose of a FC is to connect different layers. Whereby, the ANN combines each node in a layer to nodes in another layer widening the variety of attributes. This makes the CNN more capable of classifying images.

5.3.3 DL Buzzwords

5.3.3.1 Padding

It is a process to symmetrically add extra element/s around the input map. It holds a main role in a CNN, allowing to adjust the size of data. This helps to overcome downsides of shrinking the output after convolution and bad image-corners resolving. Zero-padding is the most common type, where data are surrounded by zeros. In figure 5.33, for example, the output features map is shrunk which is considered as data-loss. In comparison, figure 5.38 displays the convolution layer applied on a padded map. This results in an output map having the same size as the original input.

5.3.3.2 Softmax

It is a normalization function, which is used in classification problems. It calculates the probability of belonging to each class for a given input. Usually, it is implemented as a final-output.

5.3.3.3 Training-validating-testing phases

The pipeline of building a DL algorithm can be divided in three main phases, namely training, validating, and testing. The training phase is to train the model, where the model learns, understand, and discovers. Testing is the standardization phase to evaluate and test the learner before the real utilization. The validation phase is

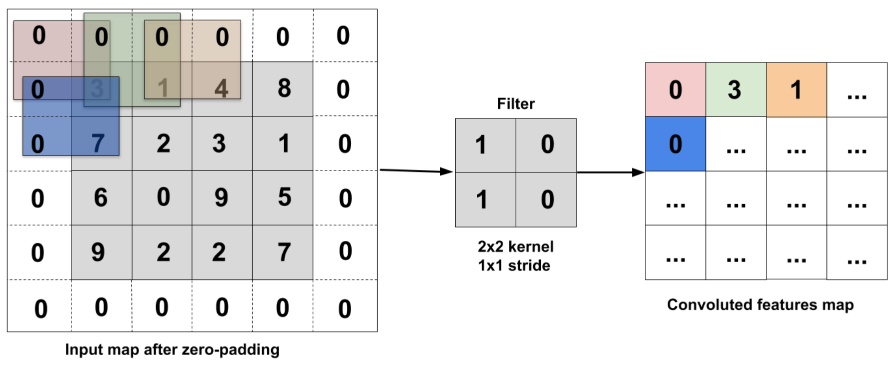


FIGURE 5.38: A convolution layer scheme with zero-padding technique. The original features map has a size of $(4x4)$ and the padded one is $(6x6)$. The applied filter layer has kernel size of $1x1$ and stride of $2x2$. The corresponding features map has a size of $(4x4)$ which is the same as the original input.

necessary to evaluate the hyperparameters of the model. Technically, the learner never learns during the validation process.

5.3.3.4 Transfer Learning (TL)

TL is a method where to share functionality and learning between models. It allows to reuse a pre-trained model as the starting point for a another model, developed for a different related task.

5.3.4 Application of DL in events classification from pnCCD datasets

5.3.4.1 Introduction

This part presents a DL approach focusing on the events reconstruction from datasets of synchrotron radiation based on Laue diffraction experiment using a white X-ray beam and an energy-dispersive 2D detector (pnCCD). This approach is able to predict the single photon's *pattern type*, referred hereafter as *event*, collected during the EDLD experiments. The model used in this task employs the softmax function at the output layer. It is equipped by a convolutional neural network (CNN) with multiple layers between the input and output layers which is mostly applied to analyze visual images. More than 70000 single photon impact events under supervised classification have been simulated and extracted to build the training-validation datasets for the algorithm.

5.3.4.2 The objective

As explained in subsection 2.3.3.1, one of the main data-preparatory steps is the single-photon event reconstruction. Its purpose is to ensure that each recorded event is generated by a single photon. In this context, the term *"fall-out event"* has been introduced. Event reconstruction and resolving of the *"fall-out events"* in real-time is a challenging task because of the huge volume of data collected during the experiment. Although the implementation of parallel programming technology using Graphic Processing Units (GPUs) to realize the reconstruction of events has proven to be a reliable technique in the case of single-photon events reconstruction (as shown in subsection 4.5.4.2), it fails to offer a solution for fall-out events. Therefore, an innovative approach is demanded to achieve a precise classification of patterns not only for single-photon events but also for *"fall-out"* events. State of the art DL-based application is introduced to perform such a task efficiently. It can precisely achieve the basic single-photon classification. Furthermore, this approach serves as a seed for the reconstruction of the fall-out events.

5.3.4.3 The model

The huge amount of available frame-data makes it possible to design an DL model, having an *end-to-end* (e2e) fashion [156]. However, a decision was made to divide the model into sub-blocks, where each block is responsible for a specific task. Figure 5.39 shows the blocks diagram of the applied DL-based model. The first two blocks are for streaming, they assign the events within a frame image. Each recognized event is sliced and sent as a 9×9 image for further processing. Since the detector size is 384*x*384 pixels, a single frame image is, in fact, a sparse matrix with only a few events. Therefore, this step is implemented to avoid the large sparsity of the frame image during the training and prediction phases. The 9×9 size was chosen based on the fact that the probability to have a *fall-out* event with spreading area greater the 9×9 is very low and can be ignored [40]. The event classifying (named *"Events-Classifier"*) and resolving (named *"Events-Picker"*) represent the AI core, where the assigned events are classified into five classes (e.g. single, double, triple, quadruple, and fall-out). Only the fall-out events are fed to the resolving block for reconstruction. The picking algorithm is an object recognition procedure, where the fall-out image is addressed into the different valid event patterns.

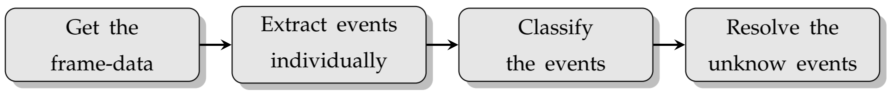


FIGURE 5.39: The blocks diagram of Dl-based model for event reconstruction.

5.3.4.4 Data preparation

a) Events-Classifier In order to assure the quality of the training procedure for the *"Events-Classifier"*, two datasets were used in combination: namely the simulated and experimental samples. The simulated dataset was generated based on constraints and conditions defined by a typical EDLD experiment, including all permutations of the events as shown in figure 5.40. In this manner, a dataset comprising 52000 events was simulated, with the events randomly distributed across different x, y pixel positions on the detector. The experimental dataset consists of more than 18000 labeled events, collected from various experiments which were performed at X-ray facilities (i.e. DELTA, EDDI, ESRF, etc.). The simulated and experimental datasets were used to feed the module during the training-validation phase, after combining and shuffling them. To test the response of the trained module, only labeled experimental datasets taken from several experimental setups were utilized. This allows to ensure the reliability of the module for any experimental data, regardless of the experimental conditions.

b) Events-Picker A dataset of 500 different fall-out examples was generated, where each valid example was annotated by an anchor box.To train a robust classifier, experimental and simulated data were considered as the data-sources. To achieve a fast annotation, a free open source tool was used, namely *"LabelImg"* [157]. Figure 5.41 shows two examples of the annotated *fall-out* event images. Each event-type is surrounded by a colored box. The positions and labels of the boxes are saved and used as a part of the training data. Another dataset was prepared for the testing procedure. It contains 200 pre-defined fall-out events examples.

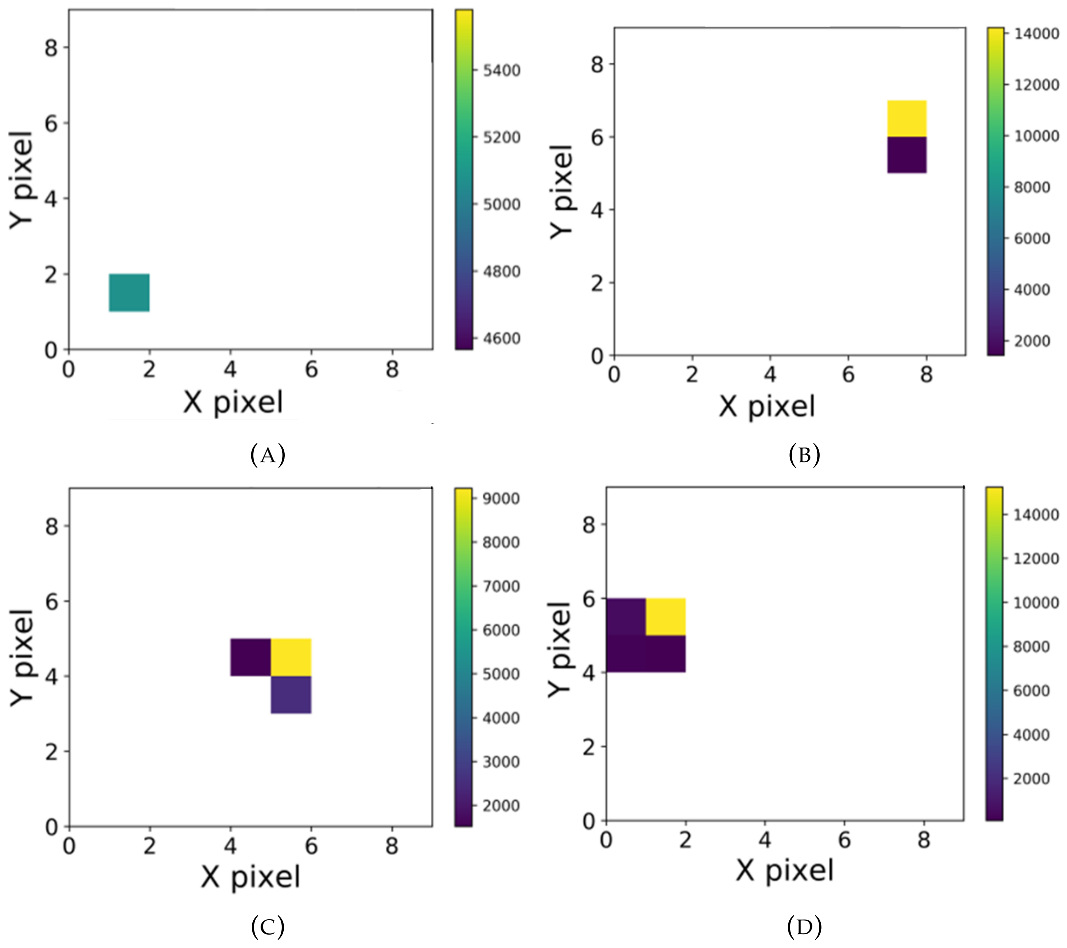


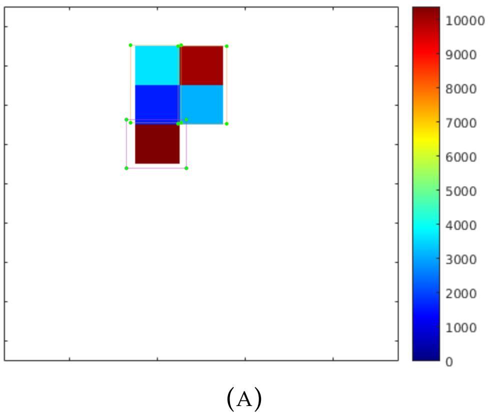
FIGURE 5.40: Randomly chosen events from the simulated dataset. Single, double, triple and quadruple events are shown correspondingly in (A), (B), (C), and (D).

5.3.4.5 Network Architecture

a) Events-Classifier The Events-Classifier architecture is shown in figure 5.42. It is composed by an input layer followed by three 2D-convolutional (Conv2D) layers with rectified linear unit (ReLu) activation function, each one is connected with a Max pooling layer. The output of these layers is flattened and sent to a fully connected layer with the softmax activation function output layer, which contains five classes. In order to boost the performance of the CNN, the ReLu is coupled with dropout that has a rate of 35%.

The hyper-parameters (i.e. the numbers of layers, the number of neurons in each layer, the size and number of filters, the stride size, the rate of dropout, the learning rate, the optimizer ...etc) have been selected by trial and error method after performing many testings of architecture versions.

The events image data are streamed as a 3D sparse-matrix with $9 \times 9 \times 3$ elements



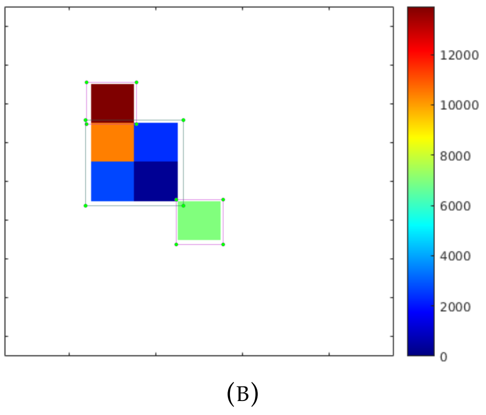


FIGURE 5.41: Randomly chosen annotated fall-out events using *"LabelImg"*. The color-bar is shown here only for better visualization. (A) is a single and two double events. (B) shows a combination of two singles and a quadruple.

representing the X, Y sizes and 3 RGB-channels for each element, respectively. The images are streamed as patches to the network, giving an output as probabilities of one of the five classes indicating the event type (single, double, triple, quadrapole or unknown). The features of *Keras library* with *TensorFlow* back-end have been implemented for the network designing . The data were split into two sets (i.e. the training and validation sets) with a splitting ratio of $90 - 10\%$. The training was executed using a machine with the specifications provided in table 4.1. The network filter coefficients have been trained using the sparse-categorical cross-entropy loss function optimization and a learning-rate of 0.001. *Adam optimizer* has been used to optimize the cost function, with the default values offered by the author, namely $\beta 1 = 0.9$, $\beta 2 = 0.999$ and $\epsilon = 10^{-8}$ [158]. The model was fitted with 30 (see figures 5.43a and 5.43b) and 100 (see figures 5.43c and 5.43d) epochs, with the patch size of 128. The accuracy and loss curves for different number of epochs are shown in figure. 5.43. As it is shown in both modules, the weights convergence within the first 30 epochs following the exponential decay function. Figure 5.43 displays linear converge behavior between 30 and 100 epochs. For more than 100 epochs, the validation loss starts to diverge, which is an indication of over-fitting. Thus, a decision was taken to have 100 epochs as a maximum value for the training-validation phase. Figure 5.44 shows the summary of the layers architecture and the parameters. At the end of the training-validation phase, two trained-modules have been generated for the testing phase, namely *Events-Classifier30* and *Events-Classifier100*.

b) Events-Picker The Events-Picker is an object detection algorithm based on *"You Only Look Once (YOLO)"* neural network [159]. YOLO is a real-time object detection which has 24 convolutional layers followed by 2 fully connected layers. Figure 5.45 visualizes the original scheme of the YOLO architecture [159]. The classes and training-data sets were customized to fit the required application. All the customizations and configurations were done by using the features abstracted in "ImageAI" library [160]. The annotated dataset containing the anchor boxes details was spitted into training and validating sets with a ration of $90 - 10\%$, respectively. The model was trained for 100 epochs with a patch size of 8.

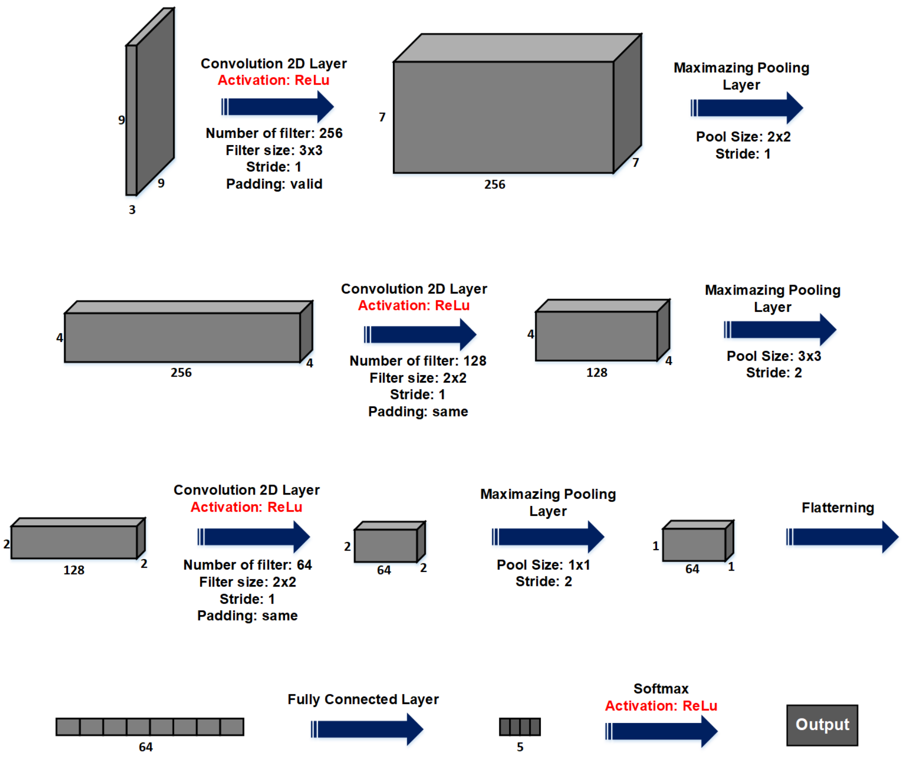


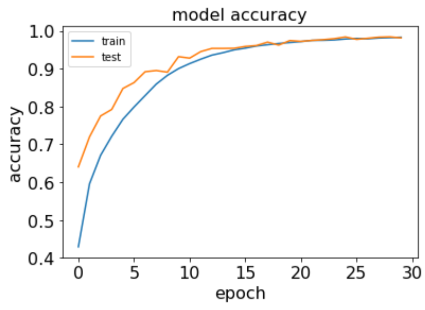
FIGURE 5.42: The convolutional neural network (CNN) with input layer, three pairs of convolutional and pooling layers, a fully connected layer, and an output layer.

5.3.4.6 Results and discussion

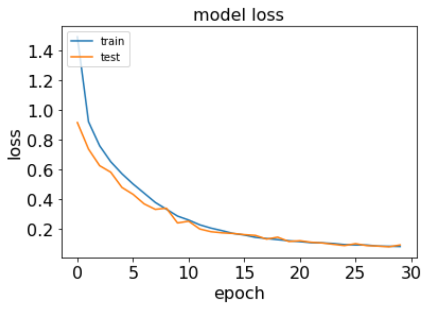
Events-Classifier30, *Events-Classifier100*, and *Events-Picker* trained-modules have been tested with the corresponding testing dataset. The prediction accuracy has been calculated, as follows:

$$\beta = \frac{n_{cor}}{n_{all}} \times 100\% \tag{5.14}$$

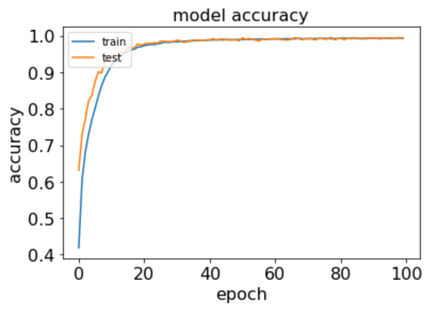
where β is the prediction accuracy, n_{cor} is the total number of the correctly predicted events, and n_{all} is the total number of events in the testing dataset.
The module Events-Classifier30 shows the average prediction accuracy of $\approx$ 92%, while the module Events-Classifier100 is able to reach more than 99% of the prediction accuracy. As a result, the CNN correctly predicts the events patterns and categorizes the streamed events in five different classes, namely single, double, triple, quadruple and fall-out events. All in all, Events- Classifier100 provides a reliable approach that can be used for further implementation. Both models give execution time of 3.9 ± 0.03 sec per 5×10^4 event ($\approx$ 167 frame). However, this time is quite long compared with the HPC approach (see section 4.5.4.2) which can perform the same data size in 1.1 ± 0.1 sec.
However, the *Events-Picker* shows a poor prediction accuracy with score of $\approx$ 62%. It was suggested that this behavior might be a result of the small training data size or the used ANN architecture. Therefore, a decision was made to assign this as an open-topic for further work.
Altogether, this part introduced a collective summary of the fundamentals and the commonly used terms in DL. It demonstrated that the usage of CNNs is a suitable



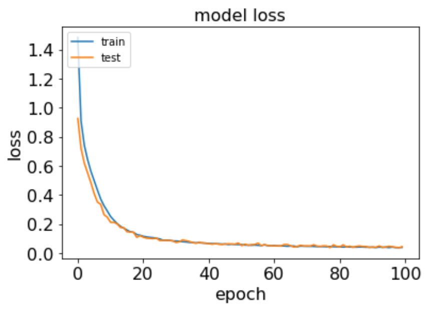
(A)



(B)



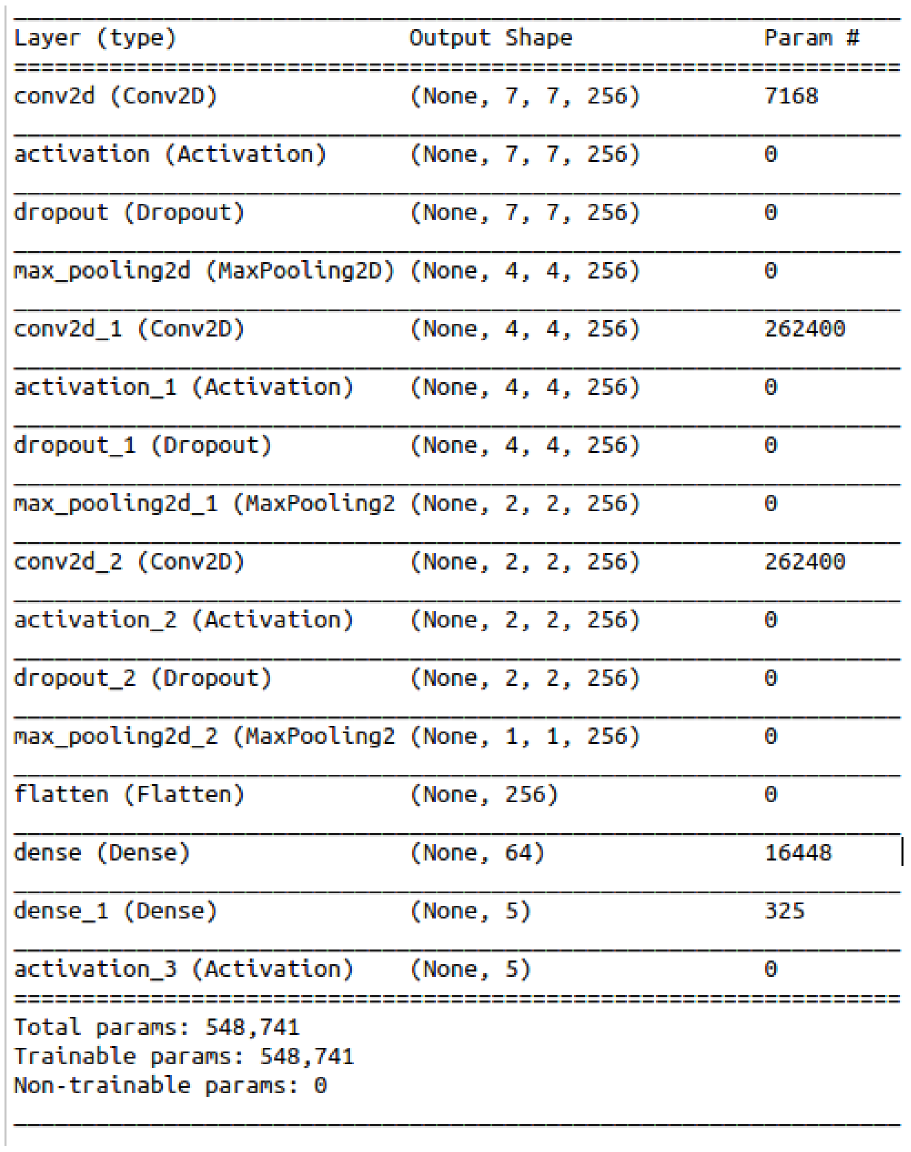
(C)



(D)

FIGURE 5.43: The accuracy and loss curves. (a) and (b) are for 30 epochs while (c) and (d) are for 100 epochs. The blue and orange lines are corresponding to the training and the validation phases, respectively.

approach to solve the event reconstruction problem. It proved that implementation the right architecture and hyper-parameters might enhance the model accuracy. It also gave an example to re-cast and implement a pre-trained model.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 7, 7, 256)         7168
_________________________________________________________________
activation (Activation)      (None, 7, 7, 256)         0
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 256)         0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 4, 4, 256)         262400
_________________________________________________________________
activation_1 (Activation)    (None, 4, 4, 256)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 256)         0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 2, 256)         262400
_________________________________________________________________
activation_2 (Activation)    (None, 2, 2, 256)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2, 2, 256)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 64)                16448
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 325
_________________________________________________________________
activation_3 (Activation)    (None, 5)                 0
=================================================================
Total params: 548,741
Trainable params: 548,741
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 5.44: Layers and parameters of the network architecture.

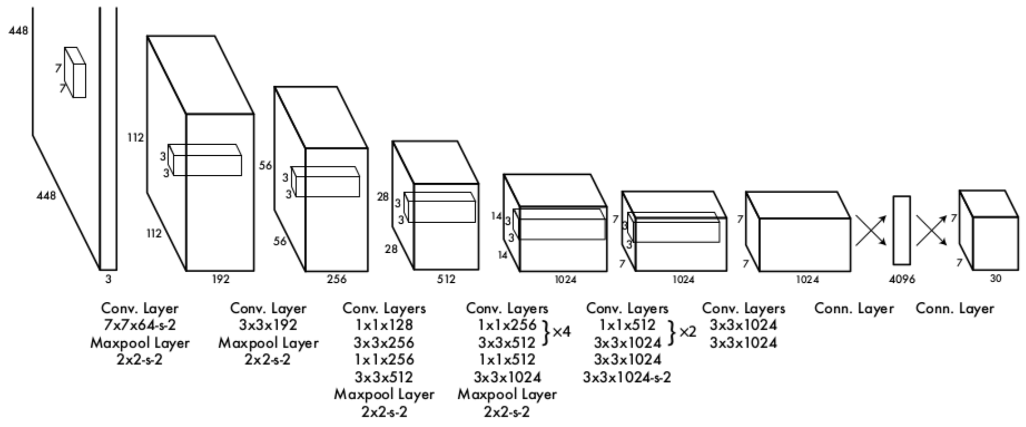


FIGURE 5.45: The original scheme for the YOLO object detector. It has 24 convolutional layers combined with max-pooling layers. It is equipped by 2 fully connected layers before the output layer [159].

Chapter 6

Application of EDLD in material texture analysis

"As we continue to improve our understanding of the basic science on which applications increasingly depend, material benefits of this and other kinds are secured for the future."- Henry Taube

6.1 Introduction

Polycrystalline materials, such as metal and ceramics, are formed from combination of many single crystalline grains. Each grain provides a certain distribution of orientations with respect to an external reference system of coordinates. The degree of the preferential grain orientations in space is characterized by the term "texture." Determination of this orientation distribution relative to a selected reference frame enables a description of the texture. With this in mind, the question arises: why do we care about texture?
Material processing technologies (e.g. welding, rolling, forging, annealing, solidification), predominantly, affect the texture of the materials, which has a significant influence on their macroscopic properties. Studying how texture changes under these processes is referred to as *material-evaluation*, which provides insight into the history of the material and enhances its manufacture to achieve an optimization of certain properties. Undoubtedly, it is essential to investigate a material's microstructure to acquire a deeper understanding of its properties, which is necessary for the practical use of this material in engineering applications.
Many techniques and research projects, such as Electron Back-Scatter Diffraction (EBSP) [161], Slip Trace Measurement (STM) [162], Close Spaced Sublimation (CSS) [163], have been applied to investigate the Micro-Texture (MiTx). Although these approaches have shown reliable results, they provide rather limited solution to at least one of the methodological challenges related to the MiTx exploration, namely; experimental setup complicities (i.e sample rotation, experiment components alignment, etc) and Data management (i.e. pattern overlap, statistical corrections, etc)
Due to the advantage of the EDLD in measuring simultaneous position- and energy-resolved signals and the new data management approach, this chapter demonstrates application of EDLD in MiTx characterization of polycrystalline materials. Employing synchrotron X-ray imaging, an energy-dispersive 2D detector and high performance computing, the MiTx investigation can be achieved *on-the-fly* by *one-shot experiments*. It is organized so that, section 6.2 explains the crystallographic concerns involved in this work. Section 6.3 demonstrates the engaging of the EDLD in MiTx analysis, including experimental concern,analytical procedure, and the results. The

developed tools is descried in section 6.4, while section 6.5 is devoted for the conclusion.

6.2 Crystallography behind the scene

The physical and mechanical properties of single crystals vary along different crystallographic directions (anisotropy). For example, the shear behavior of graphene differs with respect to the loading direction. In other words, the maximum shear failure strain of graphene in one direction is 70% higher than that in the opposite one [164]. This applies to several other physical properties, such as thermal expansion coefficient, heat transfer coefficient, and dielectric coefficient [165]. IA textured polycrystalline aggregate (i.e. having a preferred orientation) may also show direction-dependent properties. These aggregated properties tend to average out all microscopic properties of individual grains within the material and may have higher or lower degree of macroscopic anisotropy [166, 167]. Monitoring and controlling the texture is requested to predict anisotropy of materials. X-ray methods are non-destructive techniques to observe and determine the nature of the texture.

6.2.1 X-ray methods

Diffraction patterns of single or multi-crystalline materials have the form of isolated Laue spots(as explained in chapter 2), while the pattern of randomly oriented polycrystalline materials (powder) are enclosed in concentric intensity-uniform Debye-Scherrer rings. Considering these two cases as extremes, the diffraction patterns of textured materials represent an intermediate case. They show non-uniform (for slightly textured specimen) or discontinuous (for highly textured) intensity distribution along the circumference of the Debye-Scherrer rings. Figure 6.1 shows a continuum of diffraction patterns, the source of the image is [168]. It displays three different diffraction patterns behaviors: figure 6.1a shows a pattern of a single ideal crystal has isolated Laue spots. Figure 6.1b

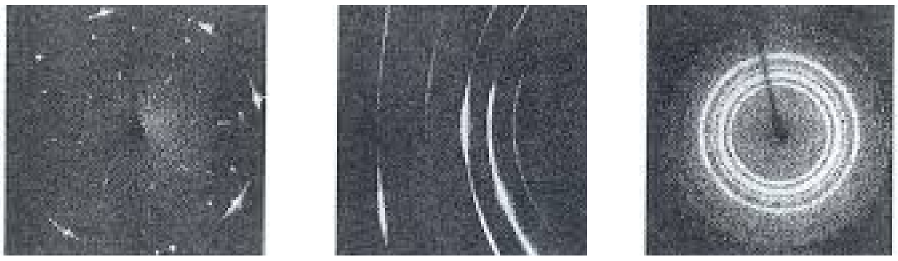

(A) (B) (C)

FIGURE 6.1: A continuum of diffraction patterns, source [168]. (A) is for a single ideal crystal, while (B) represents the diffraction pattern of a textured material. (C) is the Debye-Scherrer rings of powder diffractionn.

For monochromatic X-ray diffraction (a fixed wavelength beam), each ring is an indicator of diffractions corresponding to a certain diffraction angle (e.g. a certain d-value). In addition, according to Bragg's law (c.f. equation **??**), each ring is a representation of a specific family of crystallographic planes within the reciprocal space. Analysis of these rings and their non-uniformity might be treated as a conclusive determination of the type and degree of the texture occurrence [167]. However, the

fixed wavelength beam limits the number of crystal planes involved in the same diffraction pattern. Thus, a lot of effortful experimental procedures (e.g. sample rotation and beam manipulation) as well as statistical corrections (e.g. synoptic pattern reconstruction and defocussing correction) have to be executed to fulfill an extensive observation of the texture.
In contrast, white X-rays (a ranged wavelength beam) exploit the energy degree of freedom, enabling the inclusion of a huge range of d-value allowing to involve several crystal planes into a single *energy-dispersive diffraction* pattern. In fact, the generated pattern is nothing but an aggregation of all monochromatic diffraction patterns generated within the accessible energy range. The main challenges of this technique are data management, processing complexities and involvement of a special kind of detectors (i.e. energy-dispersive detector).

6.2.2 Texture representation

Broadly speaking, texture evaluation and description is a complex process. Therefore, several mathematical and graphical techniques are used in fundamental research and industrial application to represent textures.

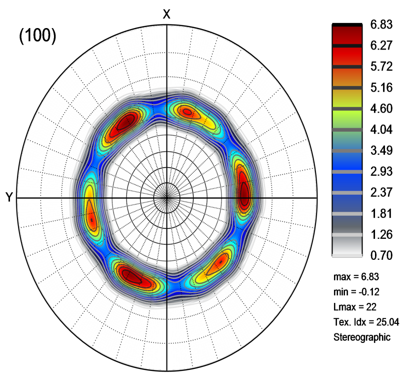
a) Mathematical representation: The representation of orientation distribution is commonly employed in a quantitative description of the texture. The notation used to describe the texture is as follows [169]:

$$\text{Aggregate Texture} = \sum_j w_j . \{hkl\}_j . \langle uvw \rangle_j \tag{6.1}$$

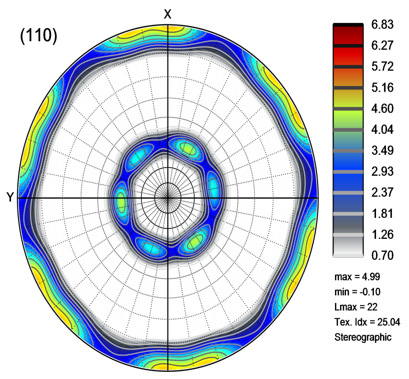
where $\{hkl\}_j . \langle uvw \rangle_j$ is *the jth lattice plane and its orientational component* [170] and w_j is weighting factor representing the probability of *the jth main orientation component*. In essence, this notation means the $\{hkl\}$ crystallographic planes and directions $\langle uvw \rangle$ in the most of grains are oriented almost parallel to texture plane and texture direction, respectively. This orientation is referred to as *preferred orientation*. A textured material may have one or more preferred directions due to the formation mechanism. Indeed, there are many other texture representation methods (e.g. Orientation distribution function (ODF)). The detailed overview of ODF calculation can be found in [167, 169].

b) Graphical representation: According to the stereographic projection, the space of all possible scattering vectors defines a sphere in the sample coordinates, centered on the diffraction volume. Many pieces of information about the sample can be plotted as scalar fields on this sphere. Since it is impossible to distinguish the positive and negative directions of the crystallographic plane norms, the data plotted on the sphere correspond to an unsigned norm, known as *"pole"*. In other words, these spheres are centrosymmetric, allowing to analyze just one hemisphere. The 2D projection of a hemisphere is said to be a pole figure.
For experiments using a point detector, a pole figure has projection area of a hemisphere with the center point corresponding to a specific crystallographic set. To plot one pole figure, the sample is rotated along the loading axis (Y- axis) and the transverse axis (X- axis). This procedure is repeated, while changing the detector position to generate a sequence of pole figures corresponding to different crystallographic planes. Figure 6.2 shows two different simulated pole figures for a textured

Aluminum specimen, where figure 6.2a shows (100) pole and 6.2b is (110). The sample and the laboratory coordinates were defined to be identical. These figures were produced using Analysis Tools for Electron and X-ray diffraction (ATEX) software package [171].
For experiments using position sensitive detectors (i.e CCD and image plates) [172], a pole figure has the projection area of a hemisphere with the incident beam at its center. To collect data orthogonally and azimuthally rotated along the prescribed orientation, the sample is rotated along the loading axis (Y- axis) and the transverse axis (X- axis) respectively. Comparably, using an energy-dispersive point detector with a white beam, it is possible to extract the pole figure by rotating the sample along the loading axis [25] or by utilizing many detectors simultaneously [26].



(A)



(B)

FIGURE 6.2: Simulated pole figures of a textured specimen of aluminum, where X-axis is the transverse axis and Y-axis is the loading axis. (A) is the texture direction of (110), while (B) is for direction (100).

6.3 EDLD for texture analysis

This section demonstrates the employing ofthe EDLD in MiTx analysis. Given the angle- and energy-dispersive nature of EDLD, the method is advantageous for texture investigation by one-shot experiments. As opposed to other methods, EDLD makes it possible to avoid the experimental intricacies, mentioned above. EDLD-MiTx is based on utlization of a white X-ray beam combined with a 2D ed detector (pnCCD). The detector active area can then cover any convenient range of diffraction angle (θ), depending on the geometry parameters (i.e. SSD, detector size, and other). The 5D characteristics of such a detector exploits the advantage to observe a complete diffraction spectra in any fixed direction. However, a drawback of the use of EDLD arises, namely a huge workload to analyze pattern overlap, which requires ultra-efficient techniques to extract the final output.

6.3.1 Understanding the texture form an Energy-dispersive Laue pattern

The texture influence on the energy profile of the recorded Laue pattern can be summarized into two scenarios, as follows:

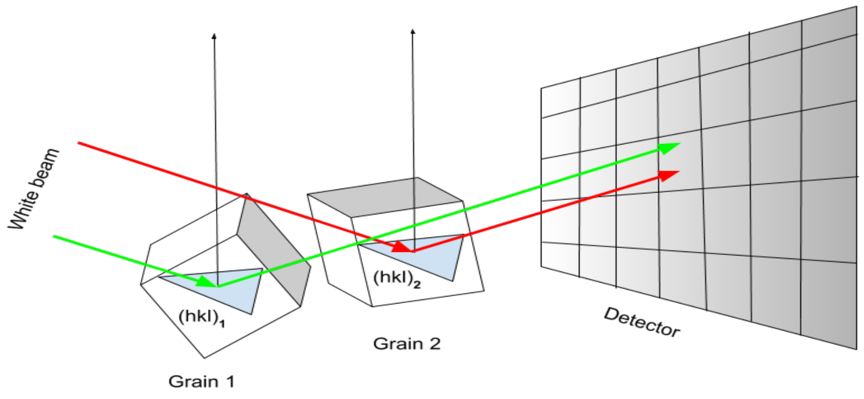
- Multi-peaks spectrum: According to Bragg's law, recording several energy values at a certain Bragg angle is an indicator for simultaneously collected multiple diffractions assigned to the different crystallographic planes, which fulfill

Bragg's condition. These planes should have parallel norms and be related to different grains. Figure 6.3a schematically demonstrates a 3D visualization of this scenario, where two different parallel planes, $(hkl)_1$ and $(hkl)_2$, are probed by a polychromatic beam. The diffracted beam includes two different energy values at the same point.

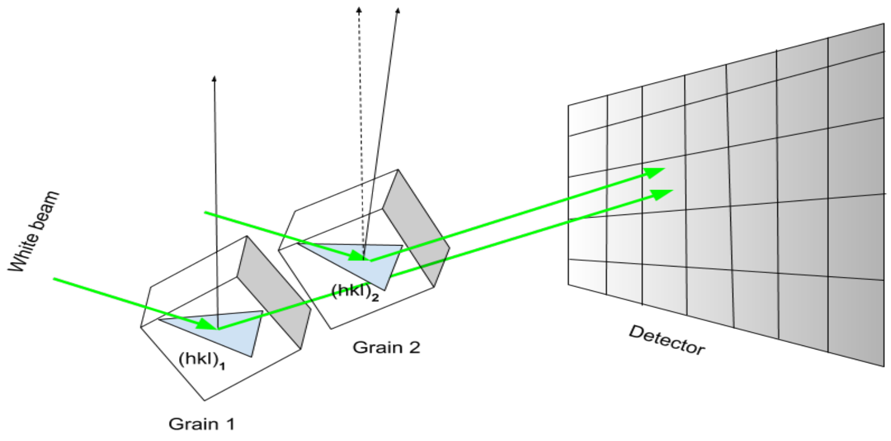
- Peak broadening: Generally speaking, the spectral broadening of the Laue diffraction occurs when the tensile loading is applied. This denotes that the macroscopic texture is changed due to defect accumulation [173, 174]. Tilting between grains within the same cluster (i.e. having the same preferred orientation) produces variation in d-spacing between their identical orientation-family planes, illuminated by the incident beam. Figure 6.3b shows the 3D scheme of this scenario, where two grains from the same cluster have their $(hkl)_1$ and $(hkl)_2$ to be nearly parallel. The collected rays demonstrate slight energy shifting, which gives rise to the widening of the recorded energy peak.

6.3.2 Experimental procedure and data management

In the case study, a dog bone shaped aluminum alloy was previously processed by Ultrasound Cycle Fatigue machine with 5×10^6 cycles, $\epsilon = 8.8e^{-4}$ and $\sigma = 130MPa$, a detailed description of studies about this fatigue mechanism can be found in [175–178]. The sample was exposed to a white synchrotron beam with photons energies between about 8 and 100keV, supplied by BESSY-II synchrotron storage ring. The typical EDLD experimental geometry was used, with $SDD = 74 \pm 5mm$ (see chapter 2). The diffracted rays were collected by the pnCCD. The recorded raw dataset was then streamed and reconstructed by EDLD-tool, as explained in chapter 4. The formed energy-dispersive diffraction pattern is expected to induce an overlap of several inhomogeneous Debye-Scherrer rings. Figure 6.4a visualizes the recored *energy-dispersive diffraction* pattern, while figures 6.4b, 6.4c, 6.4d and 6.4e are the energy-filtered patterns produced by the *"energy filter"* module which illuminates events with energy value in a specific range defined by the user.
Technically, handling these overlapping patterns is a problematic issue, as each pixel within the detector active area is considered to be an area of interest. For simplicity, each pixel plays a role as an ED point detector enclosing the energy variation of a diffracted ray, corresponding to a specific diffraction angle. Thus, the in-memory database used for typical EDLD experiments fails to warehouse data of EDLD texture experiments. Instead, a new 3D in-memory database with data cube architecture [66, 67] has been implemented, where each pixel is represented as a cell in the cube. Figure 6.5 shows a scheme of the implemented database. The X and Y vectors specify the pixel position at the detector area, while the parameters vector demonstrates a data structure, containing all details of a pixel (e.g. energy spectrum, q-vector, 2θ, Miller indices, etc). This architecture shows a high degree of efficiency regarding data accessibility and application of generic algorithms.
It is important to mention that implementation of data cube architecture may give rise to two technical issues. 1) Resources consumption: 384×384 structure (equal to the detector size) is memory-consuming and may result in memory leaks. Moreover, for pixel-wise independent calculations, it may induce high execution time latency (for serial algorithms) or GPU resources consumption (for parallel algorithms). 2) Low quality of energy spectrum: restrictions on a shot exposure time (typically ≈ 10 min) and a large number of grains illuminated in the same shot reduce the quality of the energy spectrum recored by the individual pixel. Consequently, it is very



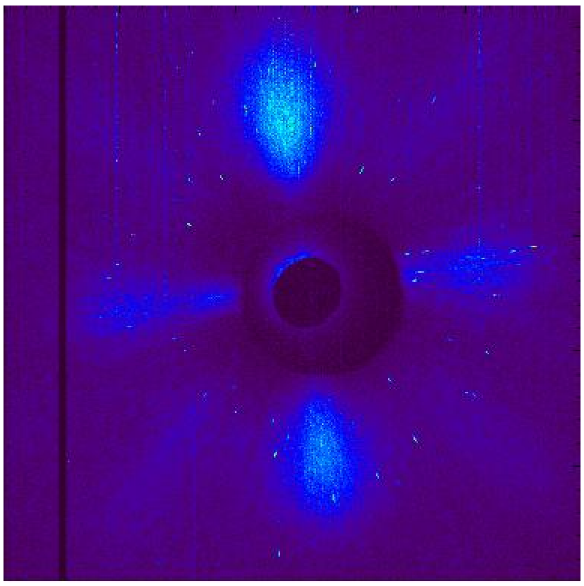
(A)



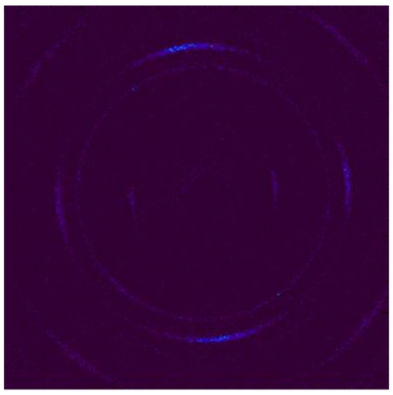
(B)

FIGURE 6.3: A scheme of two different diffraction scenarios. (A) shows two parallel planes related to two grains with different orientation with respect to the laboratory coordinates. These planes are eliminated with white incident beam, generating two different energy peaks within the same pixel. (B) displays two nearly parallel grains. The reflected beam by the same orientation planes results in energy peak broadening.

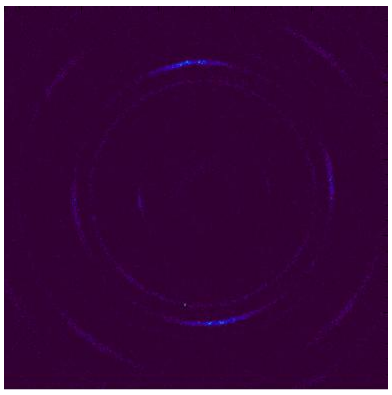
difficult to achieve automation of energy spectrum resolving (see chapter 4). Thus, a decision was taken to implement an additional step, namely image reduction and convolution, whereby each 2D neighboring pixels within a predefined mask are integrated and convoluted in one structure point. This structure is labeled as *"conV point structure"*, with implementation header file shown in Appendix B.3. This technique reduces the image size by a factor of $(1 - \frac{384 \times 384}{m_x \times m_y})$, where m_x and m_y specify the mask size, allowing to increase the energy spectrum resolution and enhance the analysis performance. However, it may also diminish the spatial certainty of the analysis. This drawback can be overcome by an optimization of either the mask size or the 2θ angles, as explained later. Figure 6.6 displays the energy spectra in both cases, where figures 6.6a and 6.6b are the original and the integrated spectra respectively.



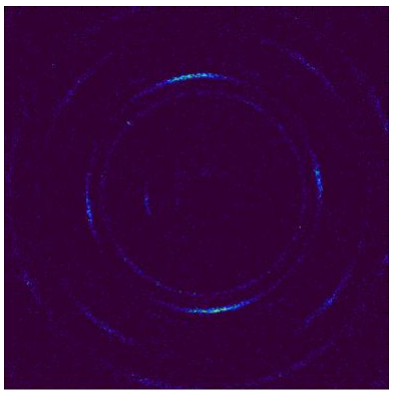
(A)



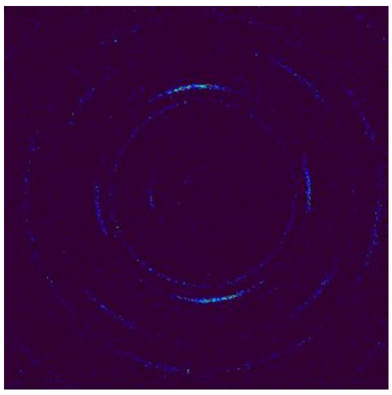
(B)



(C)



(D)



(E)

FIGURE 6.4: The diffraction patterns of a textured specimen of aluminum under different energy conditions. (A) represents the integrated intensity pattern with energy range of 8 : 100 keV. (B), (C), (D), and (E) are the filtered images with energy ranges of 43 ± 0.2, 48 ± 0.2, 54 ± 0.2, and 59 ± 0.2 keV, respectively.

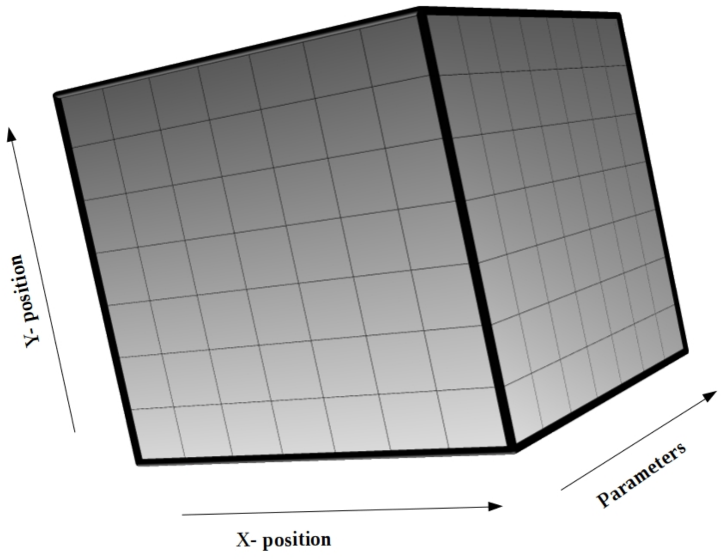


FIGURE 6.5: The in-memory cube database.

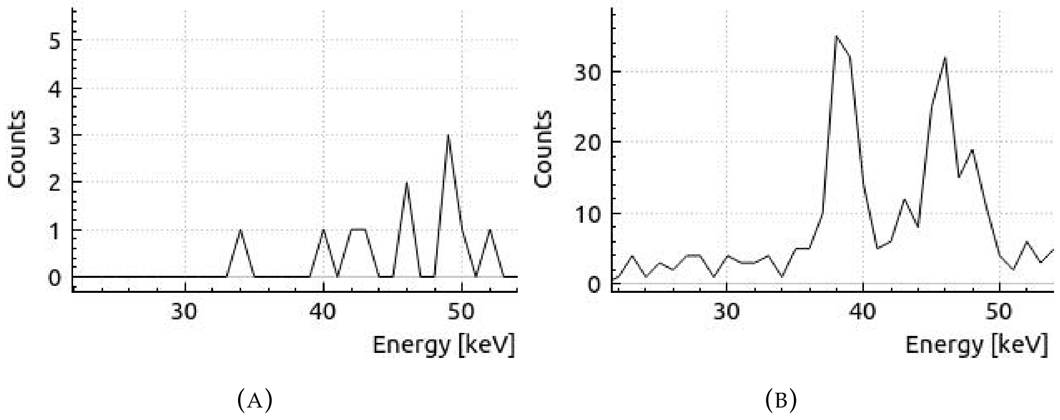


FIGURE 6.6: The energy spectra of the chosen pixel. (A) is the energy spectrum without masking. (B) is the energy spectrum with the 10 × 10 mask. The mask was done after considering this pixel to be the center.

6.3.3 Analytical procedure

Figure 6.7 shows the analytical process undertaken. The first two blocks are the pre-processing steps, whereby the data streaming path (i.e. from a prerecorded file or from the streaming module) is initialized and the hyperparameters (i.e. geometrical parameters, mask size, etc) are specified. According to these parameters, the masked intensity image is produced and visualized. Figure 6.8 exhibits the masked intensity images, given different predefined mask sizes. Subsequently, the cube database prototype, having the size of the masked image, is defined to be filled.
Calculating the point-wise parameters, such as energy spectrum, energy dominant-peak/s values, diffraction angle (θ) and diffraction angle optimization range ($\delta\theta$), is the third step. This step is performed after the application of the quantum efficiency

correction. Another parameter is introduced to define the contribution margin of the energy peak(s), namely the *"energy-weight (w_E)"*. It can be calculated using the following equation:

$$w_E = \int_{\text{FWHM}} \frac{dI}{dE} \div \int_E \frac{dI}{dE} \tag{6.2}$$

where, I is the intensity (counts), E is the energy and FWHM is the full width at half maximum corresponding to a resolved energy peak. The *"energy-weight"* is used to calculated the weighting factor shown in equation 6.1.
Aiming to assure the precision and reduce memory usage, a validity checking step is carried out, whereby a twofold test is undergone on each point. To pass this test, the intensity of each point should be above a predefined threshold and at least one energy value should be localized within the expected energy range. The points which do not meet these prerequisites are removed from the memory database.
A transformation from the lab system to the reciprocal space is performed next, following equation C.9. Miller indices can be calculated, based on equation C.14, having considered the structure factor and the standard deviation of measured Bragg angles and energies, as explained in chapter 2. Points, reveal only one valid indexation set, are denoted as *"Valid-Points"*. Multi-indexation points (i.e. having several energy peaks) are donated as *"Accumulative-Valid-Points"* and their indexation sets are vectorized and treated as several *"Valid-Points"* indexation sets. In order to reduce spatial uncertainty, this procedure is performed with θ-optimization over the predefined range $\pm\delta\theta$, in which the diffraction angle is treated as a continuous parameter instead of a discrete parameter. The optimization function is executed over the θ range with precision up to 0.01°. The optimal solution, which fulfills the optimization conditions, is stored in the *"conV point structure"* for further steps.
Finally, the points are clustered into families, depending on the corresponding orientation. Each family represents a pole or a main orientation component. The weighting factor (w_j) is calculated as follows:

$$w_j = \frac{\sum_{P_{hkl}} w_E}{\sum_{P_V} w_E} \tag{6.3}$$

,where P_{hkl} is the number of the valid points related to a specific hkl and P_v is the number of all valid points.

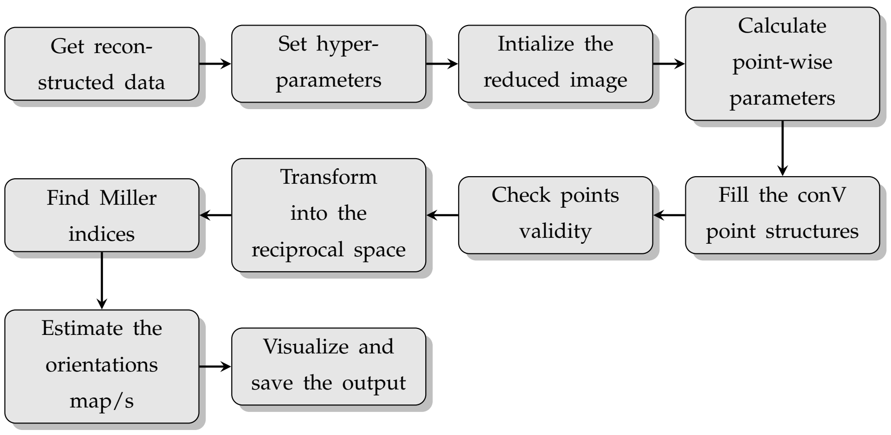
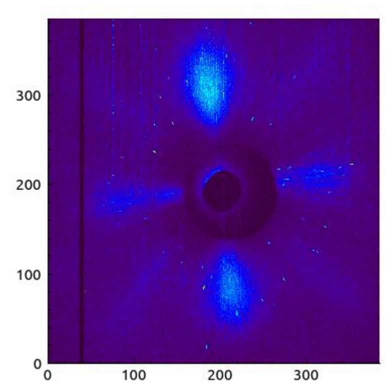
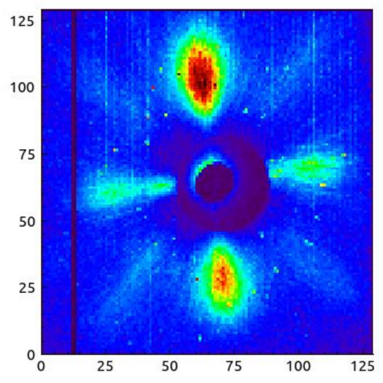

Get recon-structed data
Set hyper-parameters
Intialize the reduced image
Calculate point-wise parameters
Fill the conV point structures
Check points validity
Transform into the reciprocal space
Find Miller indices
Estimate the orientations map/s
Visualize and save the output

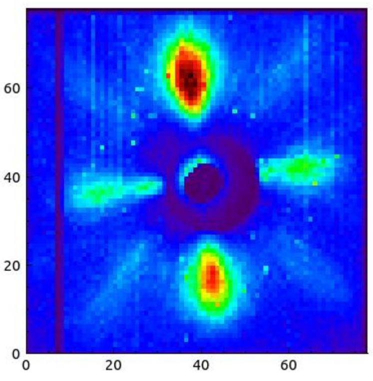
FIGURE 6.7: Analysis procedure pipeline.



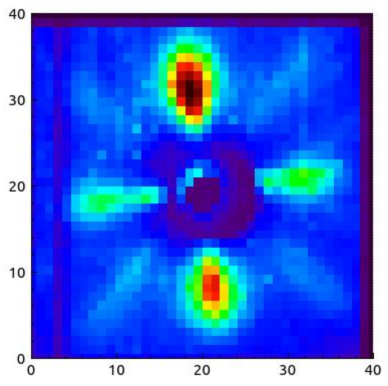
(A) 1×1



(B) 3×3



(C) 5×5



(D) 10×10

FIGURE 6.8: The masked intensity images for different mask sizes.

6.3.4 Results and discussion

6.3.4.1 Observations

As expected, different energy distribution behaviors were observed. Figure 6.9 shows the energy spectra of four randomly chosen points. Figures 6.9a, 6.9b, and 6.9c show the multi-peaks spectra, indicating the contribution of different interplanar spacing (d) for the same diffraction angle (2θ). Figure 6.9d displays a single peak spectrum, where only one reflection has been resolved.

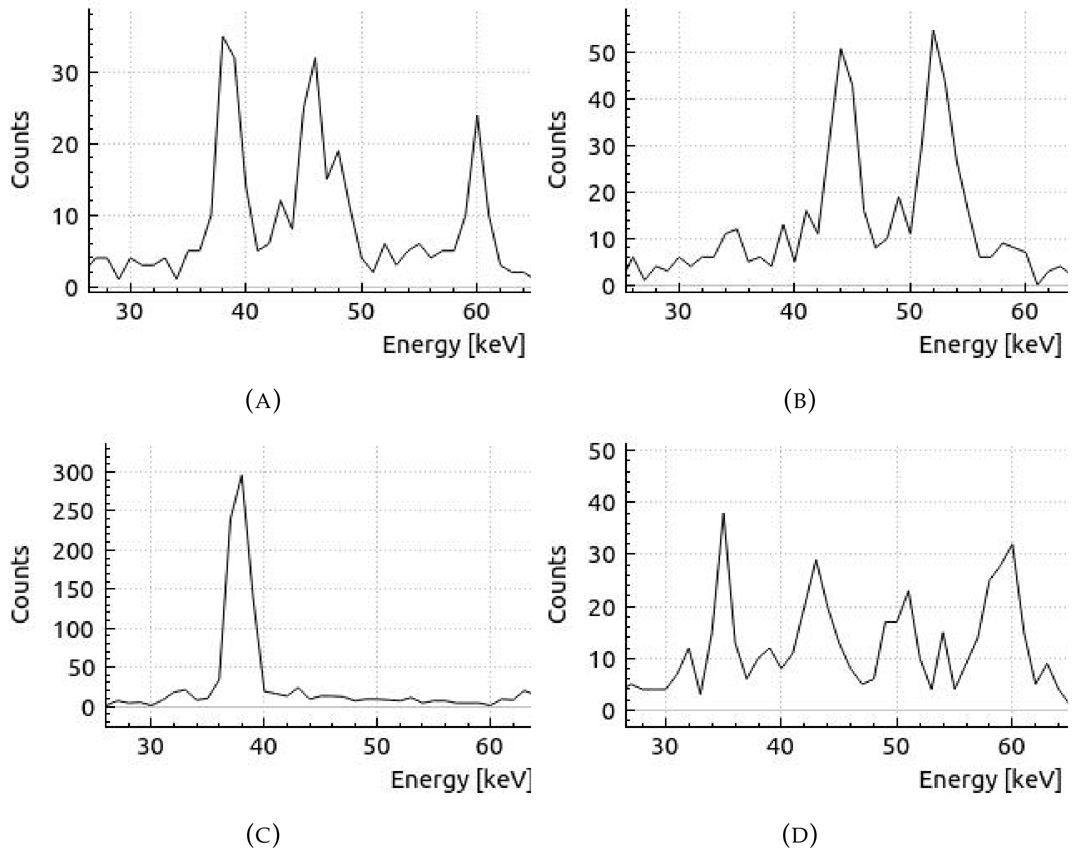


FIGURE 6.9: Energy spectra for randomly chosen masked points.

6.3.4.2 Mathematical representation

As the final output of the analytical procedure, five grains clusters have been estimated according to the following weighted orientation description;

$$\begin{aligned}\text{Aggregate Texture} =& 0.26 \pm 0.02 \times \{113\} + \\ & 0.26 \pm 0.02 \times \{111\} + \\ & 0.25 \pm 0.02 \times \{002\} + \\ & 0.19 \pm 0.02 \times \{022\} + \\ & 0.02 \pm 0.02 \times \{133\}\end{aligned} \tag{6.4}$$

In line with equation 6.4, the $\{113\}$, $\{111\}$ and $\{002\}$ orientation families can be described as the most preferred orientations, representing $\approx$ 75% of the illuminated grains. 19% of the grains are oriented in such a way that their $\{022\}$ orientations

families are parallel or near parallel to the texture plane. {133} shows a small contribution, which can be neglected.
Ignoring the orientation of the sample with respect to the laboratory coordinates impedes the direct description of lab-wise directions. Therefore, the directions description is downgraded to be sample-wise, defining 3D rotation angles with respect to the a chosen direction family. A decision was taken to choose the direction family of the most preferred orientation as the preferential direction family. Drawing upon this approach, the final description of the texture formation is described as follows;

$$\begin{aligned}\text{Aggregate Texture} =& 0.26 \pm 0.02 \times \{113\}\ (0^\circ) + \\ & 0.26 \pm 0.02 \times \{111\}\ (22^\circ) + \\ & 0.25 \pm 0.02 \times \{002\}\ (76.74^\circ) + \\ & 0.19 \pm 0.02 \times \{022\}\ (49.54^\circ) + \\ & 0.02 \pm 0.02 \times \{133\}\ (25.94^\circ)\end{aligned} \tag{6.5}$$

,where the first component is the preferential component, given $\langle 331 \rangle$ as the comparing direction family. Grains corresponding to {111} and {002} clusters show equal preferentiality degree of $\approx$ 26%. They have 3D the rotation angles of 22° and 76.74° between the comparing direction and their $\langle 111 \rangle$ and $\langle 002 \rangle$, respectively. Grains with preferred orientation of {022} show lower contribution ($\approx$ 19%) with 3D orientation angle of 49.54°. Only few grains ($\approx$ 2%) have {133} as a preferred orientation, which gives a 3D angle of 25.94°.

6.3.4.3 Graphical representation

Figure 6.10 shows a 3D model of the unweighted distribution of the five determined orientation families in reciprocal space. It displays the binary regression (e.g. valid or not valid) of the reciprocal space, where only the valid points are considered neglecting the wighting factor contribution. The two-dimensional projection of the weighted three-dimensional data is shown in figure 6.11. In principle, this 2D-projection is considered as the global pole figure representing the five different orientation families.

6.4 Technical description of the tool

The tool is an object-oriented program based on the concept of classes inheritance, in which all classes are inherited from the base class. The base class has the designing pattern of abstraction class, with the implementation header file shown in B.4. It employs many class libraries, such as ROOT-CERN [88], Eigen [90] and QCustomPlot. The software package is written in C/C++ and CUDA [92] with a user interface designed in Qt cross-platform [93].
The independence nature of the data (pixel-wise) makes it possible to implement parallel programming techniques. Moreover, the 3D mapping was performed by using a high performance graphics liberary, namely OpenGl [179].

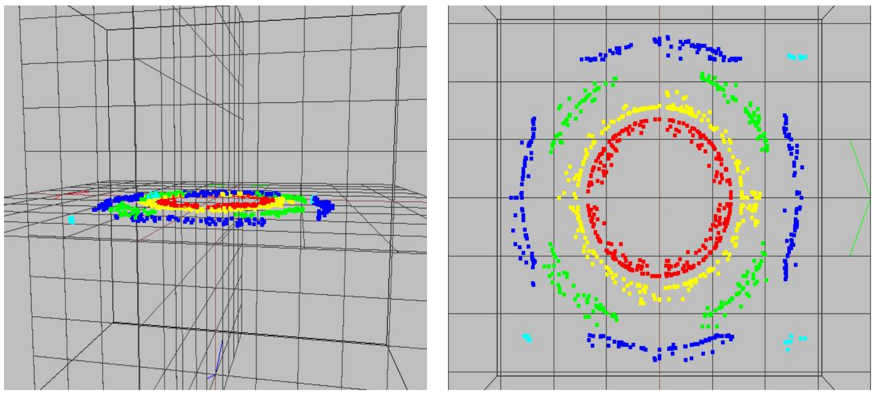


FIGURE 6.10: Binary regression 3D mapping of the reciprocal space. The red, green, blue cyan and yellow points represent the $\vec{q}$ vector associated to {111} , {022}, {113}, {133} and {002}, respectively. (A) is the side view (i.e. perpendicular to y-z plane) while, (B) is the plane view (i.e. perpendicular to x-z plane) with the incident beam at its center.

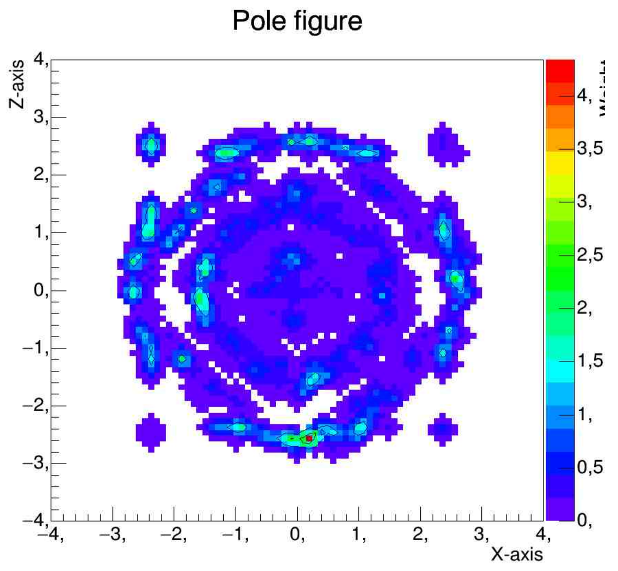
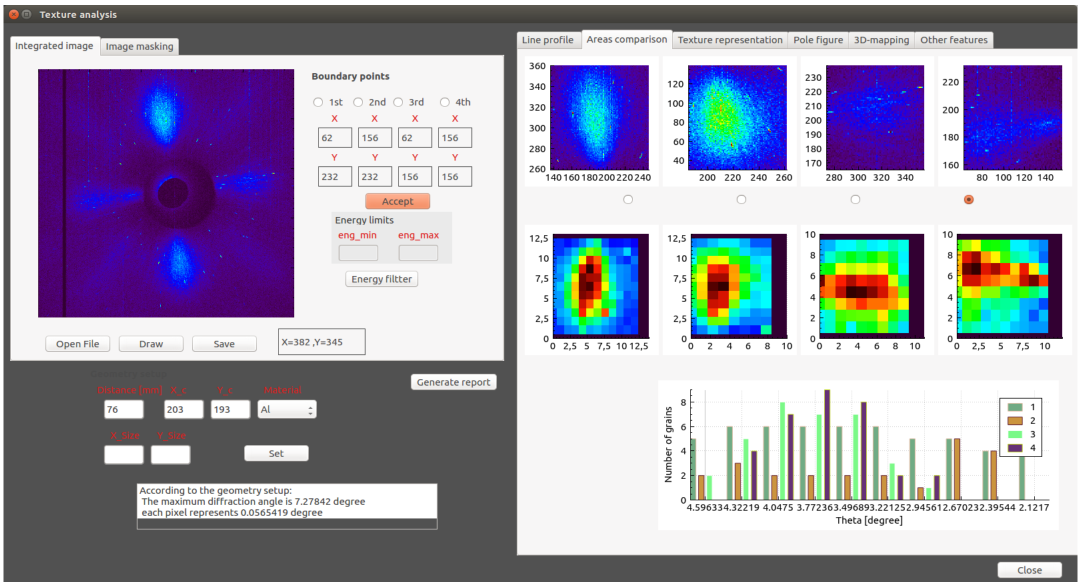


FIGURE 6.11: The final weighed 2D-projection (Pole figure).

6.5 Conclusion

This chapter demonstrated how EDLD might be applied as a non-destructive method to resolve the propagation of texture presented in materials. It was shown that EDLD is the optimal solution to avoid the experimental complexities, whereby a huge reciprocal volume is covered within the same pattern. Moreover, due to the 3D nature of EDLD collected data, a direct representation of pole figure is achievable without any sample rotation.

It was proven that the use of the in-memory database with the cube architecture enhances data accessibility and warehousing. Furthermore, *on-the-fly* analysis is possible by the implementation of the HPC and parallel computing techniques.



	1st	2nd	3rd	4th
X	62	156	62	156
Y	232	232	156	156

X=382 ,Y=345

Distance [mm]	X_c	Y_c	Material
76	203	193	Al

X_Size	Y_Size

According to the geometry setup:
The maximum diffraction angle is 7.27842 degree
each pixel represents 0.0565419 degree

FIGURE 6.12: The Graphical User Interface (GUI) of the used tool

Chapter 7

Summary and conclusion

This work explored different trend technologies, offering a collective summery of each single procedure. It presented a simple mini-review of each discipline that can be used by scientists regardless their technical experience and background. In addition, a preparatory introduction about the different X-rays sources and the detectors industry was included. A special method of the X-ray diffraction, namely: Energy Dispersive Laue Diffraction (EDLD) was investigated, indicating its cons and pros. Moreover, it was emphasized that EDLD can serve as a reliable and effective NDT method provides information that can fertilize the development in several scientific areas. From a conceptual point of view, it could be shown that the 2D energy dispersive detectors (e.g. pnCCD) enable the 5D resolution (including the intensity) detection of the X-rays. This helps to harvest the maximum benefits of the EDLD, achieving the one-shot experiment without any experimental complexities.
Considering datasets collected by EDLD experiments as a case study, it was proven that the implementation of the data clouding and warehousing can enrich data-handling procedure. It was demonstrated that the cloud-based storage can provide the scientists with a smarter environment to share their data. Moreover, it helps to overcome the deficiencies arise from the traditional data-transferring techniques. In addition, it suggested that utilization a SaaS cloud model is the most effective candidate, as it allows the users to control the cloud infrastructure regardless their technical skills. Upon that, an user-friendly tool has been developed authorizing the users to view, edit, and add datasets stored in the cloud. This tool was tested within a small size private cloud (i.e. 5 : 10 users) with infrastructure belongs to the department of solid state physics of University of Siegen.
On the topic of HPC, the GPU-based software package, namely "*EDLD-tool*" has been released to perform a full analysis procedure in the real-time. It is able to accomplish the demanded tasks on a scale of seconds, allowing for *on-the-fly* experiment steering. The deploying of IMDB shows a powerful data warehousing procedure, facilitating faster response times than disk-optimized databases. In order to simplify the using of the tool, EDLD-tool is equipped with a GUI platform giving the user a full control over the tool. However, many this approach showed some problems, such as the failure to reconstruct the *fall-out* events and the high latency execution time to identify the grains in a polycrystalline Laue pattern.
In the course of this work, various AI techniques were developed to settle issues emerged by the regular programming approaches. It could be proven that ML algorithms can perform the the grains identification task with high precision and less time. A new data presentation method, namely *AT-map* was introduced. It allows to rearrange the dataset in such a manner that the ML algorithm can understand. Moreover, using a combination between two different ML concepts (i.e. the hierarchical clustering and the Elbow method) can enhance the accuracy of the final output. This

algorithm was tested by two different datasets and showed agreement with the expectations.
The potential of DL application for photons events reconstruction could be demonstrated. Two NN architectures based on the CNN were developed and trained using simulated and experimental datasets. 1) The Event-Classifier which can classify the recored events with precision up to 99%. 2) The Event-Picker which acts as an object detector and has the architecture of *YOLO* neural network. It predicts the fall-out events with score up to 62%.
As a final output, all algorithms, APIs, and models were gathered into one platform named "MultiGrainAnalyzer (MGA)". It servers as a comprehensive control board, where each model can be launched by pressing the corresponding button. Figure 7.1 displays the GUI of the MGA. It shows six different buttons linked to various models, as follows:

- Run the pnCCD: it is to operate the pnCCD detector based on the module provided by the manufacturer.
- Streaming module: it launches the streaming module mentioned in chapter 4
- Cloud control: this button is connected to the the DOI generator and releasing tools that shown in chapter 3.
- EDLD data analysis: it launches the EDLD-tool mentioned in chapter 4.
- Energy filter: it launches the energy filter module mentioned in chapter 6.
- Texture analysis: it launches the texture tool mentioned in chapter 6.

The limitations of this work have been devoted to future research. First, all mentioned algorithms operate under an operating system (i.e. they are window-applications). It is suggested to develop them as web-applications, allowing for flexible access. Second, extending the applications range of the MGA to include different crystallographic fields, such as nanowires investigation, textile fibers and polymers study. Third, improving the neural network architecture to boost the accuracy of *fall-out* events reconstruction.

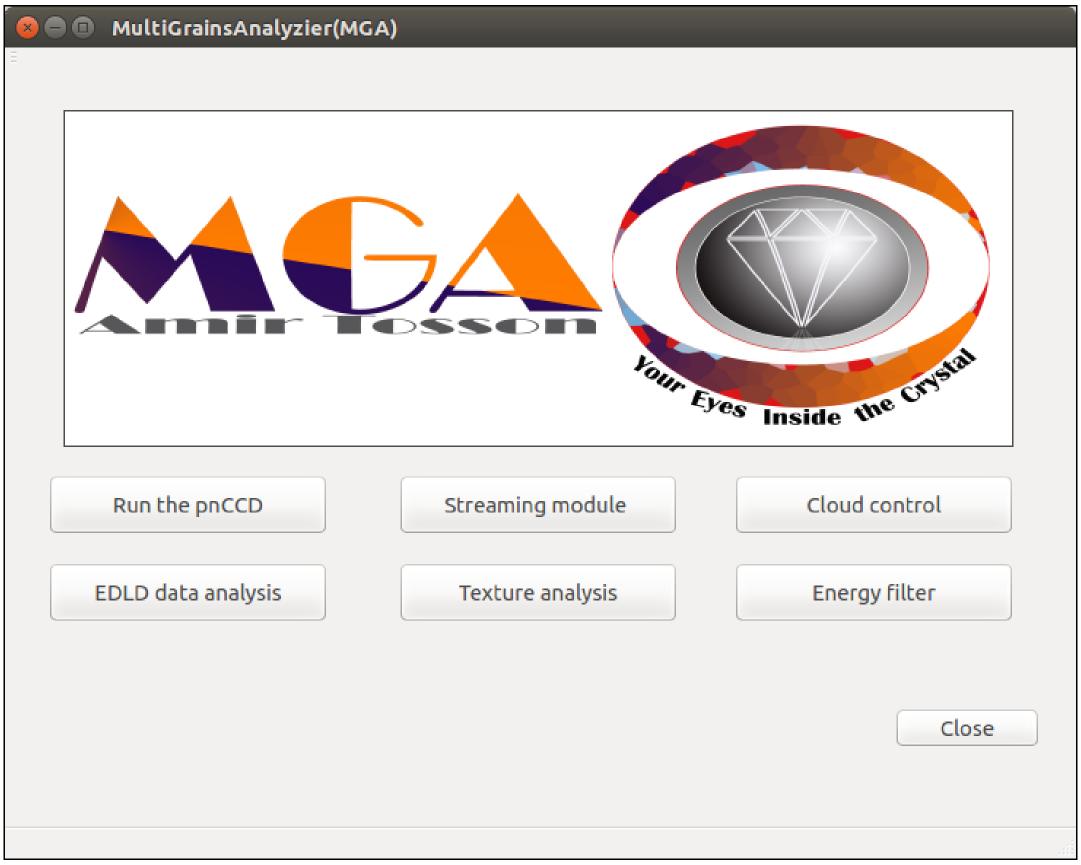


FIGURE 7.1: The main control board of the MGA platform.

Appendix A

Design patterns in software engineering

A design pattern is a template that describes a solution to a frequent problem with in a software. It is NOT the final code that can be release as a product, but it provides a tested and proven development paradigms. The main aim of these pattern Design patterns can speed up the development process by providing tested, proven development paradigms. Common design patterns can be improved over time, they can be grouped into 3 groups, as follows:

A.1 Creational design patterns

These design patterns are all about class instantiation. This pattern can be further divided into class-creation patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done. The commonly used patterns are:

- Abstraction class: It creates an explicit instance of several families of classes. It helps to improve the creation mechanism.
- Prototype: It helps to create a fully. initialized instance to be copied or cloned. It is useful if the code requires several similar objects as it allows to reuse and modify existing objects.
- Singleton: A singleton is a class of which only a single instance can exist. It may often be beneficial in case the code requires only a single instance of the class.

A.2 Structural design patterns

These design patterns are used to define the relation between the classes and the objects and organize different classes and objects, forming larger structures to obtain new functionality.

- Private Class Data: Restricts accessor/mutator access
- Adapter: it matches the interfaces between the different classes. It can convert data into several format and help objects to establish collaboration with it.
- Proxy: It is used to provde a substitude or placeholder for another object called "the intermediary", which acts as an interface to another resource.

A.3 Behavioral design patterns

These design patterns are all about the communications between the objects of classes. They provide solution regarding how the objects interact and how they can be segregated.

- Iterator: it is the simplest and the most common used pattern. Each data collection should provide an iterator, allowing the sequentially access to the elements within the collection.
- Null Object: Designed to act as a default value of an object
- Observer: The main usage is to notify any change of the state in one object to another object without keeping these two objects tightly connected.
- Visitor: Defines a new operation to a class without change

Appendix B

Codes

B.1 EDLD abstraction class header file

```
#ifndef EDLD_H
#define EDLD_H

#include <QDialog>
#include <QFont>
#include "maianalysis.h"
#include "spottrap.h"
#include "spottrapped.h"
#include "energyspectrumauto.h"
#include "matrial.h"
#include "qcustomplot.h"
#include "livestream.h"
#include "frames.h"
#include "gpuconnect.h"
#include "spotstypy.h"
#include <QInputDialog>
#include <QPainter>
#include "recomb.h"
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <QVector>
#include <QWhatsThis>
#include <QPen>
#include <QStringList>
#include <QDataStream>
#include <QProgressDialog>
#include "/usr/local/include/opencv2/core.hpp"
#include "/usr/local/include/opencv2/imgproc.hpp"
#include "/usr/local/include/opencv2/highgui.hpp"
#include "/usr/local/include/opencv2/opencv.hpp"
#include <TApplication.h>
#include <TCanvas.h>
#include <TH2D.h>
#include <TStyle.h>
#include <TColor.h>
#include <TF1.h>
#include <TF2.h>
#include <TH1.h>
#include <TMath.h>
#include <TGraph2D.h>
#include <TGraph.h>
#include <TTree.h>
#include <TGraphErrors.h>
#include <TSpectrum2.h>
#include <QPainter>
```

```
#include <algorithm>
#include <iterator>
# include </usr/include/eigen3/Eigen/Core>
# include </usr/include/eigen3/Eigen/Geometry>
# include </usr/include/eigen3/Eigen/Dense>
namespace Ui {
class EDLD;
}

class EDLD : public QDialog
{
Q_OBJECT
void mainPattern();
public:
QString filename;
QString filenameid;
QString gainfile;
QString ctefile;
spottrap ojj;
matrial lc;
spottrapped ob;
maianalysis obj;
SocketTest ok;
frames ojec;
gpuConnect gpu;
spotAll *reflection;
struct spotfin;
struct grainor;
double LC,ssd;
int val2[5];
int pro;
int pro2;
int op_max,op_min;
double op_res;
unsigned int nFound;
spotfin *m_spots;
energyspectrumauto obengspec;
int img2[384][384]={{}};
int img[384][384]={{}};
void plotImage(std::vector <int> x, std::vector <int> y);
void energySpecAll(std::vector <int> *engo);
void energySpecUsr(std::vector <int> *engall, std::vector <int>*yall,
     std::vector <int> *xall, int xusr, int yusr, double gan_fa);
void specPlot(std::vector <int> *engall, std::vector <int> *yall,
     std::vector <int> *xall, int xusr, int yusr, double gan_fa);
std::vector<int> ttt;
std::vector<int> x2_rec;
std::vector<int> y2_rec;
std::vector<int> id2;
std::vector<int> xtr;
std::vector<int> ytr;
std::vector<double> qvector;
std::vector<float> engtr;
std::vector<float> eng_kev;
std::vector<double> twotheta;
std::vector<double> dvalue;
std::vector<int> h;
std::vector<int> k;
std::vector<int>l;
std::vector<float>stdev;
std::vector<int> eng2_rec;
std::vector<int>live;
int int_img[384][384];
explicit singanalysis(QWidget *parent = 0);
~EDLD();

private slots:
void on_load_clicked();
void on_draw_clicked();
void on_eng_clicked();
void on_take_clicked();
void on_spottrap_clicked();
void on_see_clicked();
void on_eng_2_clicked();
void on_gain_clicked();
void on_cte_clicked();
void on_theta_clicked();
void on_index_clicked();
void on_anglec_clicked();
void on_az_clicked();
void on_loadrec_clicked();
void on_allch_clicked();
void on_serpara_clicked();
void on_opt_stateChanged(int arg1);
void on_pushButton_2_clicked();
void on_save_clicked();
void on_saveeng_clicked();
void on_spottrap_2_clicked();

private:
Ui::EDLD *ui;
};

#endif // EDLD_H
```

B.2 Pixel structure header file

```
#ifndef PIXELSTYPE_H
#define PIXELSTYPE_H
#include <vector>
typedef struct singlePixel {
singlePixelt(){        // constructor
engEvents.clear();
engEvents.push_back(0);
}
std::vector<unsigned int> engEvents;    // energy spectrum
int xpix,ypix;                          // pixel positions
unsigned int intensity;                 // intensity

}pixMap;
extern pixMap singlePixelt;

#endif // POINTTYPE_H
```

B.3 conV point structure header file

```
#ifndef CONV_PIXEL_H
#define CONV_PIXEL_H
#include <vector>
typedef struct convPoint {
        convPoint():intensity(0),energy(0),qx(0),qy(0),qz(0),dval(0),
        d_hkl(0),h_fin(0),k_fin(0),l_fin(0),orientationFamily{},
         direction{},hasIndextaion(false){
                eng_conv.clear();
                eng_conv.push_back(0);
        }
        double theta;                              // difraction angle
        unsigned int energy;                       // dominant energy value
        std::vector<unsigned int> eng_conv;        // energy spectrum
        int xpix,ypix;                             // pixel positions
        int h_fin,k_fin,l_fin;                     // final indextation
        unsigned int intensity;                    // intensity
        double qx, qy,qz;                          // q-vector components
        double dval,d_hkl;                         // planar spacings
        double h, k, l;                            // real hkl
        double solid_ang_std;                      // indexation st-dev
        int orientationFamily[3], direction[3];    // orientation , direction
        bool hasIndextaion;                        // validity checker
        std::string OrKey;                         // orintation as a key

}convMap;
extern convMap pixelConv;

#endif // CONV_PIXEL_H

#ifndef POINTTYPE_H
#define POINTTYPE_H

#endif // POINTTYPE_H
```

B.4 Texture-tool abstraction class header file

```
#ifndef TEXTURETOOL_H
#define TEXTURETOOL_H

#include <QDialog>
#include <QFileDialog>
#include <QMessageBox>
#include "qcustomplot.h"
#include "pixel.h"
#include "baseclass.h"
#include "rings.h"
#include "analysisequation.h"
#include "pointtype.h"
#include <exception>
#include <vector>
#include <utility>
#include <algorithm>
#include <memory>
#include "glwidget.h"
#include "grainfinder.h"
#include <exception>

namespace Ui {
class textureTool;
}

class textureTool : public QDialog ,public analysisEquation
{
Q_OBJECT

public:
explicit textureTool(QWidget *parent = 0);
~textureTool();
```

```
QString filename;
struct maps;
struct texture_map;
void drawConvMainImg();
void drawSubImage(QCustomPlot *k, QCPColorMap*colorsub_map_1,
     maps *m, int mode,int inX);
void drawConvMap(QCustomPlot *k, maps ma , QCPBars
     *regen, QCPBarsGroup *group, int ind, QVector<QString> *barTicks);
void defineLines(QCustomPlot *k,std::vector<convPoint> *convmap,int inX);
QCPBarsGroup *group;
QPoint re_point;

private slots:
void Mouse_current_pos(QMouseEvent *evv);
void Mouse_current_pos_sub(QMouseEvent *evv);
void Mouse_current_pos_eng(QMouseEvent *evv);
void Mouse_released(QMouseEvent *evv);
void Mouse_Pressed(QMouseEvent *evv2);
void Mouse_Pressed_sub(QMouseEvent *evv2);
void on_open_clicked();
void on_draw_clicked();
void on_ring_clicked();
void on_accept_clicked();
void on_setline_clicked();
void on_fft_clicked();
void on_save2_clicked();
void on_save1_clicked();
void on_save4_clicked();
void on_engchec_stateChanged(int arg1);
void on_intchec_stateChanged(int arg1);
void on_intlog_stateChanged(int arg1);
void on_englog_stateChanged(int arg1);
void on_X_valueChanged(double arg1);
void on_Y_valueChanged(double arg1);
void on_Z_valueChanged(double arg1);
void on_R_valueChanged(double arg1);
void on_save_img_clicked();
void on_R_2_valueChanged(double arg1);
void on_clear_clicked();
void on_R_3_valueChanged(double arg1);
void on_R_4_valueChanged(double arg1);
void on_R_5_valueChanged(double arg1);
void on_R_6_valueChanged(double arg1);
void on_R_7_valueChanged(double arg1);
void on_img_line_stateChanged(int arg1);
void on_ll_stateChanged(int arg1);
void on_pushButton_clicked();
void on_thre_textChanged(const QString &arg1);
void on_t_est_clicked();
void on_set_clicked();
void on_ex_textChanged(const QString &arg1);
void on_ey_textChanged(const QString &arg1);
void on_ez_textChanged(const QString &arg1);
void on_color_stateChanged(int arg1);
void on_start_clicked();
void on_lo_stateChanged(int arg1);
void on_clear_sub_clicked();
void on_all_lones_clicked();
void on_line_log_stateChanged(int arg1);
void on_rec_clicked();
void on_hkl_clicked();
void on_pdf_creator_clicked();
void on_eng_conv_show_clicked();
void on_eng_conv_save_clicked();
void on_saveconv_clicked();
void on_hklall_clicked();
void on_draw_fam_clicked();

private:
Ui::textureTool *ui;
int validPonts;
QCPItemRect *rect;
QCPColorMap *colorMap;
QCPColorMap *convColorMap;
QCPColorMap *convFamilyMap;
std::unordered_map<std::string,int> hkl_map;
std::string element;
QCPColorMap *colorsub_map;
QCPColorMap *colorsub_map_1[4];
QCPColorMap *convSub_map[4];
QCPBars *bars[4];
int x_conv,y_conv;
std::shared_ptr<BaseClass>_base_class;
std::shared_ptr<BaseClass>_ring_class;
std::vector<convPoint> conv[4];
std::vector<convPoint> conv_Global_img;
std::unique_ptr<pointAll[]> reflection;
std::vector<std::pair<int,int> >pts;
std::vector<float >texture_angle;
int line_points[4];
float lineM,lineb;
int d_s,xcc,ycc;
bool geometry_setup_ok;
GLWidget _glWidget_;
QVector <QString> barTicks;

signals:

};

#endif // TEXTURETOOL_H
```

Appendix C

Calculations

Based on the 3D Laue pattern delivered by the detector, each reflection is defined by the energy resolution of individual Bragg peaks and their spatial resolution two position coordinates. This allows to directly determine both the wavelength (λ) and the diffraction angle (θ), following the equations C.1 and C.2 respectively.

$$\lambda = \frac{hc}{E} \tag{C.1}$$

where h is the Planck constant, c is the speed of light and E is the Bragg peak energy delivered by the detector.

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{\sqrt{x^2+y^2}}{SSD}\right) \tag{C.2}$$

where x and y are the absolute Bragg peak position on the detector plan, while SSD is the sample-to-detector distance often called *"the traveling distance"*. Therefore, interplanar spacing (d) is calculated by substituting in the Bragg condition (equation 2.1). The spacing distance is related to the linear combination of the reciprocal basis vectors via

$$d = \frac{2\pi}{hb_1 + kb_2 + lb_3} = \frac{2\pi}{G_{hkl}} \tag{C.3}$$

where G_{hkl} is the reciprocal lattice vector, while b_1, b_2, and b_3 are the reciprocal primitive basis vectors and can be calculated according to equation C.4

$$\vec{b_1} = 2\pi\frac{\vec{a_2}\times\vec{a_3}}{\vec{a_1}.(\vec{a_2}\times\vec{a_3})} \qquad \vec{b_2} = 2\pi\frac{\vec{a_3}\times\vec{a_1}}{\vec{a_1}.(\vec{a_3}\times\vec{a_1})} \qquad \vec{b_3} = 2\pi\frac{\vec{a_1}\times\vec{a_2}}{\vec{a_1}.(\vec{a_1}\times\vec{a_2})} \tag{C.4}$$

In addition to the above description, an equivalent formalism based on reciprocal-space can be used to characterize Bragg peak of energy E as a function of a scattering vector $\vec{q}$,

$$\vec{q} = \vec{k_f} - \vec{k_i} \tag{C.5}$$

where $\vec{k_f}$ is the scattered wave vector and $\vec{k_i}$ is the incident wave vector. The magnitude of these two vectors is unchanged as the it is a result of elastic reflections and is governed by,

$$\mid \vec{k_f} \mid = \mid \vec{k_i} \mid = \frac{2\pi}{\lambda} = \frac{E}{\hbar c} \tag{C.6}$$

Starting from 2.1 and putting everything together,

$$q = \frac{4\pi}{\lambda}\sin\theta = \frac{2E}{hc}\sin\theta = \frac{2\pi}{d} \tag{C.7}$$

So that,

$$\vec{q} = \vec{G_{hkl}} \tag{C.8}$$

Every Laue spot collected by the detector is characterized by the intensity, I, energy, E, and three dimensional coordinates,x, y and z

$$\hbar c\vec{q} = \hbar c \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix} = \frac{E}{s} \begin{pmatrix} x \\ y-s \\ z \end{pmatrix}, \tag{C.9}$$

where s is the distance between the sample and the Laue spot at the detector plane:

$$s = \sqrt{x^2 + y^2 + z^2} \tag{C.10}$$

The error $\sigma\vec{q}$ depends on the precision of position and energy of the spot such that:

$$\delta q_x = \frac{\mid x \mid}{s}\frac{\delta E}{\hbar c} + \frac{E}{s^3 \hbar c}[(y^2+z^2)\delta x + \mid xy \mid \delta y + \mid xz \mid \delta z] \tag{C.11}$$

$$\delta q_y = \frac{(s-y)}{s}\frac{\delta E}{\hbar c} + \frac{E}{s^3 \hbar c}[(x^2+z^2)\delta y + \mid yx \mid \delta x + \mid yz \mid \delta z] \tag{C.12}$$

$$\delta q_z = \frac{\mid z \mid}{s}\frac{\delta E}{\hbar c} + \frac{E}{s^3 \hbar c}[(x^2+y^2)\delta z + \mid zx \mid \delta x + \mid zy \mid \delta y] \tag{C.13}$$

The Miller indices can be found as follows:

$$\vec{q}.\vec{a_x} = 2\pi h \ , \ \vec{q}.\vec{a_y} = 2\pi k \ , \ \vec{q}.\vec{a_z} = 2\pi l. \tag{C.14}$$

After finding the reflections indexation, the collected Laue spots are then assigned into spot-groups. Each group represents a set of Laue spots that are reflected by different crystallographic planes belong to the same grain. To achieve that all reflections are coupled and tested, covering all possible permutations. This can be done by comparing the angles between the norms of the planes according to the following formula:

$$\cos\phi_{exp} = \frac{h_1h_2 + k_1k_2 + l_1l_2}{\sqrt{h_1^2+k_1^2+l_1^2}\sqrt{h_2^2+k_2^2+l_2^2}} \tag{C.15}$$

where the right hand side is the cosine of the angle between two pre-indexed reflections (planes), and it is often called the "*theoretical value* ($\cos\phi_{th}$)". The left hand side is the geometrical calculation of the angle, so-called "*experimental value* ($\cos\phi_{exp}$), and is calculated according to the setups. If the deviation between ϕ_{exp} and ϕ_{th} is less than a predefined uncertainty, then both reflections are considered to be belonged to the same grain.

Bibliography

1. Mayer-Schönberger, V. & Cukier, K. *Big data: A revolution that will transform how we live, work, and think* (Houghton Mifflin Harcourt, 2013).
2. Forum, W. E. *Big Data Doesn't Interpret Itself* <https://www.weforum.org/agenda/2019/04>.
3. Lamanna, M. The LHC computing grid project at CERN. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **534,** 1–6 (2004).
4. Bicer, T. *et al. Real-time data analysis and autonomous steering of synchrotron light source experiments* in *2017 IEEE 13th International Conference on e-Science (e-Science)* (2017), 59–68.
5. Jaskolski, M., Dauter, Z. & Wlodawer, A. A brief history of macromolecular crystallography, illustrated by a family tree and its Nobel fruits. *The FEBS journal* **281,** 3985–4009 (2014).
6. Hauptman, H. A. History of X-ray Crystallography. *Structural Chemistry* **1,** 617–620 (1990).
7. Wilkinson, M. D. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* **3** (2016).
8. Tosson, A *et al.* EDLD-Tool: A real-time GPU-based tool to stream and analyze energy-dispersive Laue diffraction of BIG Data sets collected by a pnCCD. *Journal of Instrumentation* **14,** P01008 (2019).
9. Tosson, A, Bahrami, D, Davtyan, A, Shokr, M & Pietsch, U. Deep learning application for events classification of energy-dispersive Laue diffraction datasets collected by pnCCD.
10. Tosson, A, Shokr, M & Pietsch, U. *Application of Cloud Computing for Big Data in the X-Ray Crystallography Community* in *Proceedings of the 3rd International Conference on Software Engineering and Information Management* (2020), 1–4.
11. Chatterjee, A. X-ray diffraction. *Handbook of analytical techniques in concrete science and technology*, 275–332 (2000).
12. Guinier, A. *X-ray diffraction in crystals, imperfect crystals, and amorphous bodies* (Courier Corporation, 1994).
13. Warren, B. E. *X-ray Diffraction* (Courier Corporation, 1990).
14. *Braggs Law* <https://upload.wikimedia.org/wikipedia/commons/2/26/Braggs_Law.svg/>.
15. Woolfson, M. M. & Woolfson, M. M. *An introduction to X-ray crystallography* (Cambridge University Press, 1997).
16. Merzbacher, E & Lewis, H. in *Corpuscles and Radiation in Matter II/Korpuskeln und Strahlung in Materie II* 166–192 (Springer, 1958).

17. Podgorsak, E. B. Basic radiation physics. *Radiation oncology physics: a handbook for teachers and students. Vienna: IAEA* **7** (2005).

18. Giacovazzo, C. *et al. Fundamentals of crystallography* (Oxford University Press Oxford, 2002).

19. Winick, H. & Doniach, S. *Synchrotron radiation research* (Springer Science & Business Media, 2012).

20. Jordan, E. C. *Reference data for engineers: Radio, electronics, computer, and communications* (HW Sams, 1985).

21. O'Shea, P. G. & Freund, H. P. Free-electron lasers: status and applications. *Science* **292,** 1853–1858 (2001).

22. Send, S. Utilization of a frame store pnCCD for energy-dispersive Laue diffraction with white synchrotron radiation (2013).

23. Rhodes, J. Radioisotope X-ray spectrometry. A review. *Analyst* **91,** 683–699 (1966).

24. Rodehorst, V. & Koschan, A. *Comparison and evaluation of feature point detectors* in *5th International Symposium Turkish-German Joint Geodetic Days* (2006).

25. Szpunar, J & Gerward, L. Energy-dispersive X-ray diffraction studies of the texture in cold-rolled alpha-beta brass. *Journal of Materials Science* **15,** 469–476 (1980).

26. Genzel, C. *et al.* Exploiting the features of energy-dispersive synchrotron diffraction for advanced residual stress and texture analysis. *The Journal of Strain Analysis for Engineering Design* **46,** 615–625 (2011).

27. Sonoda, M., Takano, M., Miyahara, J. & Kato, H. Computed radiography utilizing scanning laser stimulated luminescence. *Radiology* **148,** 833–838 (1983).

28. Amemiya, Y. *et al. Imaging Plate Detector In X-Ray Diffraction Using Synchrotron Radiation* in *X-Ray Instrumentation in Medicine and Biology, Plasma Physics, Astrophysics, and Synchrotron Radiation* **1140** (1989), 167–175.

29. Takahashi, K. *X-ray imaging by means of the photostimulable phosphor* tech. rep. (1990).

30. Henrich, B *et al.* PILATUS: A single photon counting pixel detector for X-ray applications. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **607,** 247–249 (2009).

31. Johnson, I *et al.* Eiger: a single-photon counting x-ray detector. *Journal of Instrumentation* **9,** C05032 (2014).

32. Andritschke, R., Hartner, G., Hartmann, R., Meidinger, N. & Struder, L. *Data analysis for characterizing pnCCDs* in *2008 IEEE Nuclear Science Symposium Conference Record* (2008), 2166–2172.

33. Strüder, L *et al.* The European photon imaging camera on XMM-Newton: the pn-CCD camera. *Astronomy & Astrophysics* **365,** L18–L26 (2001).

34. Leitenberger, W. *et al.* Application of a pnCCD in X-ray diffraction: a three-dimensional X-ray detector. *Journal of Synchrotron Radiation* **15,** 449–457 (2008).

35. F. Alghabi S. Send, U. S.A.A.U. P. & Kolb, A. Fast GPU-based absolute intensity determination for energy-dispersive X-ray Laue diffraction, *Ltd and Sissa Medialab srl* (2016).

36. F. Alghabi S. Send, U. S.A.A.U. P. & Kolb, A. Fast GPU-based spot extraction for energy-dispersive X-ray Laue diffraction. *Journal of Instrumentation* **9** (2014).
37. Shokr, M *et al.* Applications of a pnCCD detector coupled to columnar structure CsI (Tl) scintillator system in ultra high energy X-ray Laue diffraction. *Journal of Instrumentation* **12,** P12032 (2017).
38. Granato, S. *et al.* Characterization of eROSITA PNCCDs. *IEEE Transactions on Nuclear Science* **60,** 3150–3157 (2013).
39. Foucar, L. *et al.* Cass—cfel-asg software suite. *Computer Physics Communications* **183,** 2207–2213 (2012).
40. Abboud, A. *et al.* Sub-pixel resolution of a pnCCD for X-ray white beam applications. *Journal of Instrumentation* **8,** P05005 (2013).
41. Das, S. R. *Data science: theories, models, algorithms, and analytics* (S. R, Das, 2016).
42. Patil, D. *Building data science teams* (" O'Reilly Media, Inc.", 2011).
43. Snijders, C., Matzat, U. & Reips, U.-D. " Big Data": big gaps of knowledge in the field of internet science. *International Journal of Internet Science* **7,** 1–5 (2012).
44. Demchenko, Y., De Laat, C. & Membrey, P. *Defining architecture components of the Big Data Ecosystem* in *2014 International Conference on Collaboration Technologies and Systems (CTS)* (2014), 104–112.
45. Peng, R. D. & Matsui, E. *The Art of Data Science* <`https://bookdown.org/rdpeng/artofdatascience/`>.
46. Lyu, M. R. *et al. Handbook of software reliability engineering* (IEEE computer society press CA, 1996).
47. Leinhardt, G. & Leinhardt, S. Chapter 3: Exploratory data analysis: New tools for the analysis of empirical data. *Review of research in education* **8,** 85–157 (1980).
48. Law, A. M. *How to build valid and credible simulation models* in *2008 Winter Simulation Conference* (2008), 39–47.
49. Clayton, D. & Hills, M. *Statistical models in epidemiology* (OUP Oxford, 2013).
50. Sadiku, M. N., Shadare, A. E., Musa, S. M. & Akujuobi, C. M. Data Visualization. *International Journal of Engineering Research And Advanced Technology (IJERAT)* **2,** 11–16 (2016).
51. Parkhill, D. F. Challenge of the computer utility (1966).
52. Gorelik, E. *Cloud computing models* PhD thesis (Massachusetts Institute of Technology, 2013).
53. Subramanian, K. Public clouds. *A whitepaper sponsored by Trend Micro Inc* (2011).
54. Raza, M. *Public Cloud vs Private Cloud vs Hybrid Cloud: What's The Difference?* <`https://www.bmc.com/blogs/public-private-hybrid-cloud/`>.
55. Hurwitz, J. S., Kaufman, M., Halper, F. & Kirsch, D. *Hybrid cloud for dummies* (John Wiley & Sons, 2012).
56. Ashraf, I. An overview of service models of cloud computing. *International Journal of Multidisciplinary and Current Research* **2,** 779–783 (2014).
57. Mell, P., Grance, T., *et al.* The NIST definition of cloud computing (2011).

58. SanthoshBaboo, S & Renjith Kumar, P. Next Generation Data Warehouse and In-Memory Analytics. *International Journal of Computer Applications* **69,** 25–30 (2013).

59. Saad, Y. *SPARSKIT: a basic tool kit for sparse matrix computations* 1994.

60. Bell, N. & Garland, M. *Efficient sparse matrix-vector multiplication on CUDA* tech. rep. (Nvidia Technical Report NVR-2008-004, Nvidia Corporation, 2008).

61. *ScieBo - Die Campuscloud* <https://www.zimt.uni-siegen.de/dienste/campuscloud/>.

62. Garofalo, R. *Building enterprise applications with Windows Presentation Foundation and the model view ViewModel Pattern* (Microsoft Press, 2011).

63. Syromiatnikov, A. & Weyns, D. *A journey through the land of model-view-design patterns* in *2014 IEEE/IFIP Conference on Software Architecture* (2014), 21–30.

64. *.NET Cross-platform.* <https://dotnet.microsoft.com/>.

65. Miller, J. S. & Ragsdale, S. *The common language infrastructure annotated standard* (Addison-Wesley Professional, 2004).

66. Harinarayan, V., Rajaraman, A. & Ullman, J. D. *Implementing data cubes efficiently* in *Acm Sigmod Record* **25** (1996), 205–216.

67. Gray, J. & Reichart, D. C. *Efficient multidimensional data aggregation operator implementation* US Patent 5,822,751. 1998.

68. Kirk, D. B. & Wen-Mei, W. H. *Programming massively parallel processors: a hands-on approach* (Morgan kaufmann, 2016).

69. Russell, S. J. & Norvig, P. *Artificial intelligence: a modern approach* (Malaysia; Pearson Education Limited, 2016).

70. Davis, L. Handbook of genetic algorithms (1991).

71. *AWS Total Cost of Ownership (TCO) calculator.* <www.awstcocalculator.com/>.

72. Kuck, D. J., Grest, G. S., McKay, S. R. & Christian, W. High performance computing—challenges for future systems. *Computers in Physics* **11,** 259–261 (1997).

73. Cybenko, G. & Kuck, D. J. Supercomputers-revolution or evolution? *IEEE Spectrum* **29,** 39–41 (1992).

74. Thornton, J. E. The cdc 6600 project. *Annals of the History of Computing* **2,** 338–348 (1980).

75. Kumar, V. *Introduction to parallel computing* (Addison-Wesley Longman Publishing Co., Inc., 2002).

76. Asano, S., Maruyama, T. & Yamaguchi, Y. *Performance comparison of FPGA, GPU and CPU in image processing* in *2009 international conference on field programmable logic and applications* (2009), 126–131.

77. Faggin, F., Hoff, M. E., Mazor, S. & Shima, M. The History of the 4004. *IEEE Micro* **16,** 10–20 (1996).

78. Ramanathan, R. Intel® Multi-Core Processors. *Making the Move to Quad-Core and Beyond* (2006).

79. McClanahan, C. History and evolution of gpu architecture. *A Survey Paper,* 9 (2010).

80. Krewell, K. What's the Difference Between a CPU and a GPU. *NVIDIA Corporation) Retrieved January* **3,** 2016 (2009).

81. Kirk, D. *et al. NVIDIA CUDA software and GPU parallel computing architecture* in *ISMM* **7** (2007), 103–104.

82. Prasad, S., Gupta, A., Rosenberg, A., Sussman, A. & Weems, C. *Topics in Parallel and Distributed Computing* (Springer, 2015).

83. Banker, R. D., Datar, S. M., Kemerer, C. F. & Zweig, D. Software complexity and maintenance costs. *Communications of the ACM* **36,** 81–95 (1993).

84. Abran, A., Khelifi, A., Suryn, W. & Seffah, A. *Consolidating the ISO usability models* in *Proceedings of 11th international software quality management conference* **2003** (2003), 23–25.

85. Biolchini, J., Mian, P. G., Natali, A. C. C. & Travassos, G. H. Systematic review in software engineering. *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES* **679,** 45 (2005).

86. Martin, R. C. *Agile software development: principles, patterns, and practices* (Prentice Hall, 2002).

87. Send, S. *et al.* Characterization of a pnCCD for applications with synchrotron radiation. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **711,** 132–142 (2013).

88. *ROOT-Cern framework* <https://root.cern.ch/>.

89. *OpenCV Library* <https://opencv.org/>.

90. *Eigen Library* <http://eigen.tuxfamily.org/>.

91. *QCustomPlot widget* <http://www.qcustomplot.com//>.

92. *CUDA zone* <https://developer.nvidia.com/cuda-zone/>.

93. *Nokia, QT-Cross-platform application and UI framework* <http://qt.nokia.com/>.

94. Alghabi, F *et al.* Real-time processing of pnCCD images using GPUs (2012).

95. Alghabi, F *et al.* Fast GPU-based spot extraction for energy-dispersive X-ray Laue diffraction. *Journal of Instrumentation* **9,** T11003 (2014).

96. Ryan, C., Clayton, E, Griffin, W., Sie, S. & Cousens, D. SNIP, a statistics-sensitive background treatment for the quantitative analysis of PIXE spectra in geoscience applications. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* **34,** 396–402 (1988).

97. Tomoyori, K, Hirano, Y, Kurihara, K & Tamada, T. *Background elimination using the SNIP algorithm for Bragg reflections from a protein crystal measured by a TOF single-crystal neutron diffractometer* in *Journal of Physics: Conference Series* **664** (2015), 072049.

98. Morháč, M., Kliman, J., Matoušek, V., Veselskỳ, M. & Turzo, I. Background elimination methods for multidimensional coincidence γ-ray spectra. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **401,** 113–132 (1997).

99. Morháč, M., Kliman, J., Matoušek, V., Veselskỳ, M. & Turzo, I. Efficient one- and two-dimensional gold deconvolution and its application to γ-ray spectra decomposition. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **401,** 385–408 (1997).

100. Morháč, M., Kliman, J., Matoušek, V., Veselský, M. & Turzo, I. Identification of peaks in multidimensional coincidence γ-ray spectra. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **443,** 108–125 (2000).

101. Tomoyori, K. & Tamada, T. *New data reduction protocol for Bragg reflections observed by TOF single-crystal neutron diffractometry for protein crystals with large unit cells* in *J. Phys. Conf. Ser.* **762** (2016), 012040.

102. Mariscotti, M. A method for automatic identification of peaks in the presence of background and its application to spectrum analysis. *Nuclear Instruments and Methods* **50,** 309–320 (1967).

103. Singh, A. K. *Optimizing All-to-All and Allgather Communications on GPGPU Clusters* PhD thesis (The Ohio State University, 2012).

104. Abboud, A. *et al.* Single-shot full strain tensor determination with microbeam X-ray Laue diffraction and a two-dimensional energy-dispersive detector. *Journal of applied crystallography* **50,** 901–908 (2017).

105. Send, S. *et al.* Application of a pnCCD for energy-dispersive Laue diffraction with ultra-hard X-rays. *Journal of Applied Crystallography* **49,** 222–233 (2016).

106. Colom, R., Karama, S., Jung, R. E. & Haier, R. J. Human intelligence and brain networks. *Dialogues in clinical neuroscience* **12,** 489 (2010).

107. Turing, A. Intelligent machinery (1948). *B. Jack Copeland,* 395 (2004).

108. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **5,** 115–133 (1943).

109. Samuel, A. L. in *Computer Games I* 366–400 (Springer, 1988).

110. Mueller, J. P. & Massaron, L. *Artificial intelligence for dummies* (John Wiley & Sons, 2018).

111. *Oxford Economics Ltd.* <`https://www.oxfordeconomics.com/`>.

112. Baldi, P., Brunak, S. & Bach, F. *Bioinformatics: the machine learning approach* (MIT press, 2001).

113. Larranaga, P. *et al.* Machine learning in bioinformatics. *Briefings in bioinformatics* **7,** 86–112 (2006).

114. Cleophas, T. J., Zwinderman, A. H. & Cleophas-Allers, H. I. *Machine learning in medicine* (Springer, 2013).

115. Deo, R. C. Machine learning in medicine. *Circulation* **132,** 1920–1930 (2015).

116. Ivezić, Ž., Connolly, A. J., VanderPlas, J. T. & Gray, A. *Statistics, data mining, and machine learning in astronomy: a practical Python guide for the analysis of survey data* (Princeton University Press, 2014).

117. Ball, N. M. & Brunner, R. J. Data mining and machine learning in astronomy. *International Journal of Modern Physics D* **19,** 1049–1106 (2010).

118. Koller, D. & Friedman, N. Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series). *MIT Press, Aug* **31,** 2009 (2009).

119. Sra, S., Nowozin, S. & Wright, S. J. *Optimization for machine learning* (Mit Press, 2012).

120. Zhang, C. & Ma, Y. *Ensemble machine learning: methods and applications* (Springer, 2012).

121. Mitchell, T. M. *The discipline of machine learning* (Carnegie Mellon University, School of Computer Science, Machine Learning . . ., 2006).

122. Crisci, C, Ghattas, B & Perera, G. A review of supervised machine learning algorithms and their applications to ecological data. *Ecological Modelling* **240,** 113–122 (2012).

123. Khanum, M., Mahboob, T., Imtiaz, W., Ghafoor, H. A. & Sehar, R. A survey on unsupervised machine learning algorithms for automation, classification and maintenance. *International Journal of Computer Applications* **119** (2015).

124. Drugan, M. M. Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm and evolutionary computation* **44,** 228–246 (2019).

125. Learned-Miller, E. G. Introduction to supervised learning. *I: Department of Computer Science, University of Massachusetts* (2014).

126. Park, H. An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing* **43,** 154–164 (2013).

127. Peng, C.-Y. J., Lee, K. L. & Ingersoll, G. M. An introduction to logistic regression analysis and reporting. *The journal of educational research* **96,** 3–14 (2002).

128. Triola, M. F. Bayes' theorem. *PDF. Dostupno: http://faculty. washington. edu/tamre/BayesTheorem. pdf* (2010).

129. Freund, Y. & Mason, L. *The alternating decision tree learning algorithm* in *icml* **99** (1999), 124–133.

130. Yampolskiy, R. V. & El-Barkouky, A. Wisdom of artificial crowds algorithm for solving NP-hard problems. *International Journal of Bio-inspired computation* **3,** 358–369 (2011).

131. Yuan, C. & Yang, H. Research on K-Value Selection Method of K-Means Clustering Algorithm. *J* **2,** 226–235 (2019).

132. Shafeeq, A. & Hareesha, K. *Dynamic clustering of data with modified k-means algorithm* in *Proceedings of the 2012 conference on information and computer networks* (2012), 221–225.

133. Hamerly, G. & Elkan, C. *Learning the k in k-means* in *Advances in neural information processing systems* (2004), 281–288.

134. Shokr, M. From pnCCD to pnCCD+ CsI (Tl) scintillator: characterizations and applications (2019).

135. *SciPy library* <`https://www.scipy.org/`>.

136. *Matplotlib* <`https://matplotlib.org/`>.

137. Cherukara, M. J., Nashed, Y. S. & Harder, R. J. Real-time coherent diffraction inversion using deep generative networks. *Scientific reports* **8,** 16520 (2018).

138. Park, W. B. *et al.* Classification of crystal structure using a convolutional neural network. *IUCrJ* **4,** 486–494 (2017).

139. Agrawal, H. *et al.* Deep Learning Methods for Visual Fault Diagnostics of Dental X-ray Systems (2018).

140. Laanait, N., Zhang, Z. & Schlepütz, C. M. Imaging nanoscale lattice variations by machine learning of x-ray diffraction microscopy data. *Nanotechnology* **27,** 374002 (2016).

141. LeCun, Y., Bengio, Y & Hinton, G. Deep Learning. *Nature* **521,** 436–44 (May 2015).

142. Esteva Andre Kuprel, B. N.R.A.K.J.S.S.M.B.H.M.T. S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **542** (2017).

143. Kriesel, D. A brief introduction on neural networks (2007).

144. Nguyen, T. Total number of synapses in the adult human neocortex. *Undergraduate Journal of Mathematical Modeling: One+ Two* **3,** 26 (2010).

145. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* **65,** 386 (1958).

146. Síma, J. Introduction to neural networks (1998).

147. De Veaux, R. D. & Ungar, L. H. A brief introduction to neural networks. *Unpublished: http://www. cis. upenn. edu/˜ ungar/papers/nnet-intro. ps* (1997).

148. Dougherty, M. A review of neural networks applied to transport. *Transportation Research Part C: Emerging Technologies* **3,** 247–260 (1995).

149. Gardner, M. W. & Dorling, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* **32,** 2627–2636 (1998).

150. Montana, D. J. & Davis, L. *Training Feedforward Neural Networks Using Genetic Algorithms.* in *IJCAI* **89** (1989), 762–767.

151. Ren, J. S. & Xu, L. *On vectorization of deep convolutional neural networks for vision tasks* in *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).

152. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., *et al.* Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86,** 2278–2324 (1998).

153. Szegedy, C. *et al. Going deeper with convolutions* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), 1–9.

154. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

155. *Understanding Edge Detection: Sobel Operator* <https://medium.com/datadriveninvestor

156. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks* **61,** 85–117 (2015).

157. Tzutalin. *LabelImg* <https://github.com/tzutalin/labelImg/>.

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

159. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. *You only look once: Unified, real-time object detection* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 779–788.

160. *ImageAI, State-of-the-art Recognition and Detection AI with few lines of code.* <http://imageai.org//>.

161. Schwartz, A. J., Kumar, M., Adams, B. L. & Field, D. P. *Electron backscatter diffraction in materials science* (Springer, 2000).

162. Raabe, D, Keichel, J & Gottstein, G. Investigation of crystallographic slip in polycrystalline Fe3Al using slip trace measurement and microtexture determination. *Acta Materialia* **45,** 2839–2849 (1997).

163. Gao, J. *et al.* Dependence of film texture on substrate and growth conditions for CdTe films deposited by close-spaced sublimation. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films* **29,** 051507 (2011).

164. Yi, L. & Chang, T. Loading direction dependent mechanical behavior of graphene under shear strain. *Science China Physics, Mechanics and Astronomy* **55,** 1083–1087 (2012).

165. Huang, M., Chen, M. & Zheng, T. Direction-dependent functions of physical properties for crystals and polycrystals. *Acta Mechanica Sinica* **25,** 639–649 (2009).

166. Hammond, C. & Hammond, C. *The Basics of Cristallography and Diffraction* (Oxford, 2001).

167. Cullity, B. D. Elements of X-ray Diffraction (2001).

168. Dahms, M, Brokmeier, H., Seute, H & Bunge, H. Ouantitative Texture Analysis in Multiphase Materials with Overlapping Bragg Reflections. *This volume* (1988).

169. Suwas, S. & Ray, R. K. *Crystallographic texture of materials* (Springer, 2014).

170. COJOCARU, V. D., RĂDUCANU, D., Cinca, I. & CĂPRĂRESCU, A. TEXTURE ANALYSIS BY (110) POLE FIGURE FOR A SPD PROCESSED Ti-25Ta-25Nb ALLOY. *UNIVERSITY POLITEHNICA OF BUCHAREST SCIENTIFIC BULLETIN SERIES B-CHEMISTRY AND MATERIALS SCIENCE* **74,** 213–222 (2012).

171. *Analysis Tools for Electron and X-ray diffraction* <http://www.atex-software.eu/>.

172. Wcislak, L & Bunge, H. Diffraction Profile Pole Figures Measured with a Position Sensitive Detector. *Texture, Stress, and Microstructure* **26,** 19–38 (1996).

173. Rozaliya, B., Gene, I., *et al. Strain and dislocation gradients from diffraction: spatially-resolved local structure and defects* (World Scientific, 2014).

174. Shokr, M. *In situ observations of single grain behavior during plastic deformation in polycrystalline Ni using EDLD* (Materials Science and Engineering A, 2019).

175. Yao, W. & Guo, S. VHCF test and life distribution of aluminum alloy LC4CS. *International Journal of Fatigue* **30,** 172–177 (2008).

176. Bacher-Hoechst, M. & Issler, S. Assessment of very high cycle fatigue (VHCF) effects in practical applications. *Procedia Engineering* **66,** 26–33 (2013).

177. Alvarez-Armas, I *et al.* Growth of short cracks during low and high cycle fatigue in a duplex stainless steel. *International Journal of Fatigue* **41,** 95–100 (2012).

178. Hüsecken, A. K. Untersuchung der Auswirkung von sehr hohen Lastspielzahlen auf einen austenitisch-ferritischen Duplexstahl mittels in-situ Röntgendiffraktion an der Strahllinie BL10 an der Synchrotronstrahlungsquelle DELTA (2017).

179. *OpenGL Library* <https://www.opengl.org/>.

Acknowledgements

Finally, I would like to thank all my colleagues who contributed to the success of this thesis and all the people who supported me during this time:

- I wish to express my sincere appreciation to my supervisor, Prof. Dr. Ullrich Pietsch, who convincingly guided and encouraged me to do the right thing even when the road got tough. Without his persistent help, the goal of this project would not have been realized.
- I would like to acknowledge PNSensor GmbH and in particular Prof. Dr. Lothar Strüder and Robert Hartmann for the providing of the pnCCD system and for their technical support.
- I am grateful to my colleges in Solid State Physics group, University of Siegen: Dr. Shokr, Dr. Al Hassan, Dr. Bahrami, Dr. Davtyan, Al Humaidy and Dr. Abboud for helping during the beamtimes and for all the fun we have had.
- I am also immensely grateful to V. Pankova, Bonn University for her comments that greatly improved the manuscript.
- I would like to show my gratitude to my colleges in Conze Informatik GmbH for their support.
- I am very thankful to my family and my parents for the never-ending support.

This research was supported by Bundesministerium für Bildung und Forschung (BMBF) and Deutsche Forschungsgemeinschaft (DFG).