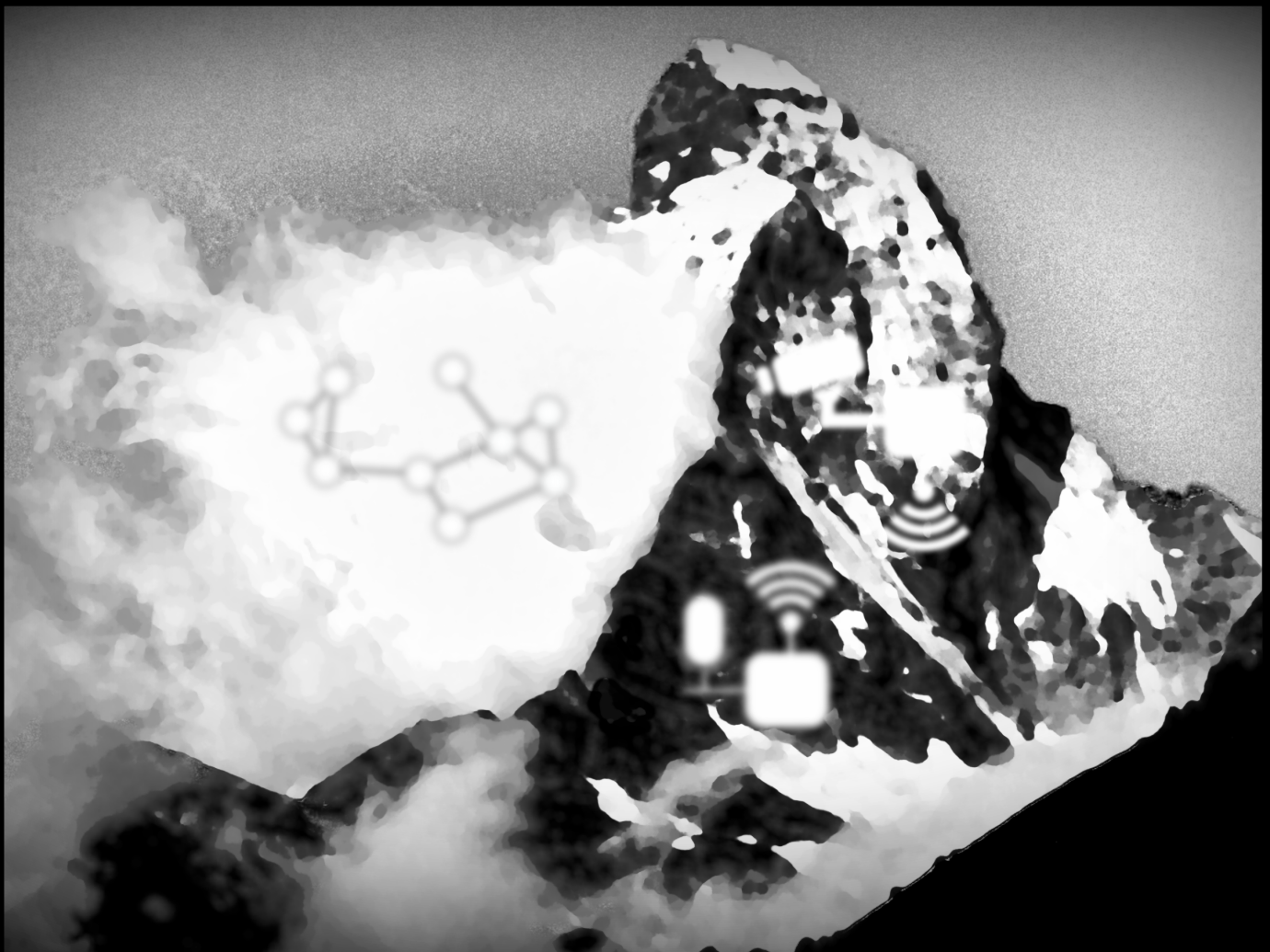


Harnessing Environmental Data at the Edge of the Cloud



Matthias Meyer

Diss. ETH No. 28024

Diss. ETH No. 28024

Harnessing Environmental Data at the Edge of the Cloud

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich
(Dr. sc. ETH Zurich)

presented by
Matthias Meyer

born on 14.07.1987
citizen of Earth

accepted on the recommendation of
Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Jan Beutel, co-examiner
Prof. Dr. Stephan Gruber, co-examiner

2021



Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

TIK-SCHRIFTENREIHE NR. 194

Matthias Meyer

Harnessing Environmental Data at the Edge of the Cloud



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A dissertation submitted to ETH Zurich
for the degree of Doctor of Sciences

DISS. ETH NO. 28024

Prof. Dr. Lothar Thiele, examiner

Prof. Dr. Jan Beutel, co-examiner

Prof. Dr. Stephan Gruber, co-examiner

Examination date: 121121

*Für meine Familien.
Die, die mich hier her gebracht haben und die, die mich weiterbringen.*

Abstract

Global warming is a defining challenge of our time with devastating consequences for local habitats. High mountain areas are particularly affected by global warming leading to a decline of their cryosphere (glaciers, snow cover and permafrost). In high-alpine steep bedrock, permafrost thaw decreases the stability of mountain slopes leading to an increase of rockfalls and landslides and thereby putting life and built infrastructure at risk.

Monitoring these environmental changes is important for natural hazard warning and understanding the geophysical processes leading to such hazards. Moreover, by providing evidence from large-scale, long-term measurements, environmental monitoring helps to bolster scientific findings and can call attention to the immediate impacts of climate change. The rise of wireless sensor networks offers a range of possibilities for environmental monitoring enabling large-scale deployments with high spatial-temporal resolution using many different sensor types. The cheap and diverse sensors can be installed at hard to reach places with little available networking or power infrastructure. However, the resulting datasets (often heterogenous and long-term measurements) require a complex data analysis. Moreover, networking or power failures often lead to an error-prone data collection and a fragmented and noisy datasets. Analyzing these datasets typically requires dedicated domain-expert knowledge which can not be scaled to long-term monitoring datasets. Machine learning provides options to extract information automatically but these techniques usually require a clean dataset for training and their performance is strongly affected by differences in the distribution of training and test data.

In this dissertation, we consequently develop tools and methods applicable to heterogeneous, long-term, noisy datasets originating in wireless sensor network deployments. The main contributions of the dissertation are

- A methodology to work with fragmented and noisy data from a real-world sensor network deployment at Matterhorn, Switzerland. The methodology uses active learning with human-in-the-loop and a heterogeneous set of sensors to systematically filter out unwanted influences from seismic signals.
- The development and installation of an array of low-power, event-triggered micro-seismic sensors for the purpose of rockfall early warning. In addition, a machine-learning based human footstep classifier is designed and optimized for computation on memory-constraint embedded devices to detect humans in the hazard zone.
- Unsupervised and semi-supervised methods designed to bridge machine learning technology and domain-expert knowledge by providing experts with

automated information extraction and machine-learning algorithms with crucial information such as information about the system context.

■ *foReal*, a data analytics and visualization platform which allows to combine data from different sources. It is designed for long-term and large-scale environmental datasets and focuses on robustness against data corruption, missing data and misconfigurations during data processing as well as misinterpretations during experiment design and analysis. The tooling developed enables fast and easy exchange between experts of various domains and offers the public access to scientific data.

Zusammenfassung

Die globale Erwärmung ist eine der entscheidendsten Herausforderung unserer Zeit mit verheerenden Folgen für lokale Lebensräume. Hochgebirgsregionen sind besonders von der globalen Erwärmung betroffen, da die Erwärmung einen Rückgang der Kryosphäre (Gletscher, Schneedecke und Permafrost) zur Folge hat. Im hochalpinen, steilen Felsgestein verringert das Auftauen des Permafrosts die Stabilität der Berghänge, was zu einer Zunahme von Steinschlägen und Erdbeben führt. Solche Bergstürze können verheerenden Auswirkungen auf Siedlungen, Infrastruktur und Menschenleben haben.

Die Beobachtung dieser Umweltveränderungen durch großflächige Langzeitmessungen ist wichtig, um vor der Naturgefahr zu warnen und um das Verständnis der geophysikalischen Prozesse, welche durch Auftauen der Permafrostböden entstehen können, zu verbessern. Außerdem können großflächige Langzeitmessungen wissenschaftliche Erkenntnisse untermauern und auf die unmittelbaren Auswirkungen des Klimawandels aufmerksam machen. Der Anstieg der Nutzung von funkbasierten Sensornetzwerken bietet eine Reihe von Möglichkeiten für die Umweltbeobachtung und ermöglicht großflächige Einsätze mit hoher räumlicher und zeitlicher Auflösung sowie vielen unterschiedlichen Sensortypen. Die billigen und vielfältigen Sensoren können an schwer zugänglichen Orten mit wenig Netzwerk- oder Energieinfrastruktur installiert werden. Die daraus resultierenden Datensätze (oft heterogene Langzeitmessungen) erfordern jedoch eine komplexe Datenanalyse. Außerdem führen Netzwerk- oder Stromausfälle oft zu einer fehleranfälligen Datenerfassung und fragmentierten und verrauschten Datensätzen. Die Analyse dieser Datensätze erfordert in der Regel Fachwissen, das sich nicht auf langfristige Beobachtungsdatensätze skalieren lässt. Maschinelles Lernen bietet Optionen zur automatischen Extraktion von Informationen. Diese Techniken erfordern aber in der Regel einen sauberen Datensatz zum Lernen und ihre Leistung wird stark durch Unterschiede in der Zusammensetzung des Trainingsdatensatzes gegenüber Daten während der Anwendung beeinflusst.

In dieser Dissertation entwickeln wir daher Werkzeuge und Methoden, die auf heterogene, verrauschte Langzeitdatensätze aus drahtlosen Sensornetzen abgestimmt sind. Die wichtigsten Beiträge der Dissertation sind:

- Eine Methodik für die Arbeit mit fragmentierten und verrauschten Daten aus einer realen Sensornetzinstallation am Matterhorn, Schweiz. Die Methodik verwendet aktives Lernen mit Human-in-the-Loop und einem heterogenen Satz von Sensoren, um systematisch, unerwünschte Einflüsse aus seismischen

Signalen herauszufiltern.

- Die Entwicklung und Installation eines Arrays von ereignisgesteuerten, seismischen Sensoren für die Frühwarnung vor Steinschlag. Darüber hinaus wird ein auf maschinellem Lernen basierender Klassifikator für menschliche Schritte entwickelt um Menschen in der Gefahrenzone zu erkennen. Dieser Klassifikator wird für die Berechnung auf eingebetteten Geräten mit begrenztem Speicher optimiert.
- Unüberwachte und halbüberwachte Lernmethoden, die maschinelles Lernen und das Wissen von Fachleuten verbinden. Dabei werden Experten mit automatische Informationsextraktion unterstützt und maschinelle Lernalgorithmen mit wichtigen Informationen versorgt, wie z. B. Informationen über den Systemkontext.
- *foReal*, eine Datenanalyse- und Visualisierungsplattform, die es ermöglicht Daten aus verschiedenen Quellen zu kombinieren. Sie wurde für langfristige und große Umweltdatensätze ausgelegt und konzentriert sich auf die Robustheit gegenüber Datenkorruption, fehlende Daten und Fehlkonfigurationen bei der Datenverarbeitung sowie Fehlinterpretationen während der Versuchsplanung und -analyse. Die Plattform ermöglicht einen schnellen und einfachen Austausch zwischen Experten aus verschiedenen Bereichen und bietet der Öffentlichkeit einen Zugang zu wissenschaftlichen Daten.

Acknowledgments

I would like to thank Prof. Dr. Lothar Thiele for giving me the opportunity and the freedom to learn and explore fields which were partially new to both of us while supporting me with insightful wisdom throughout my dissertation. I would also like to thank Prof. Dr. Jan Beutel who tied me to the rope under a helicopter and thereby introduced and guided me through the difficulties of environmental monitoring. I would like to thank my co-examiner Prof. Stephan Gruber for working through this thesis. I would like to thank all the people I had the pleasure to work with: All current and former members of TEC, my office mates, the kitchen crew. All my co-authors and collaborators also the people of the PermaSense project who I don't know, but who gave me such a strong foundation.

On the personal side I would like to thank my friends, wherever they may live now. North, south; seaside, mountains. My travel mates and long-distance friends. The ones who grew up with me, the ones who brought me through my studies (more or less) sane, doing plain stupid stuff. Our startup attempt opened the door for this work.

Last but not least I would like to thank my families, my brother and sister and my parents who paved the way so I can stand here today, who allowed me to learn and think freely. Especially, thanks to my partner who supported me throughout this time while frequently reminding me that there are nicer places than high-dimensional spaces... and cheers to my little daughter: finally I didn't spend all sleepless nights thinking about this thesis.

Table of Contents

Abstract	i
Zusammenfassung	iii
Acknowledgments	v
1 Introduction	1
2 Identification of external influences in micro-seismic recordings	13
3 Hazard monitoring using machine learning and edge devices	47
4 Enhancing domain expertise with machine learning and vice versa	81
5 foReal! Team up for real-world data exploration	103
6 Conclusions and Outlook	135
Bibliography	139

1

Introduction

Climate change challenges the earth and our society. Among others, global warming accelerates the retreat of permafrost globally [PRM⁺, BSN⁺19] acting as an accelerator for a changing world climate [MSC⁺19]. In addition to global scale effects, on local scales the thawing of permafrost has immediate consequences on our habitat. In Alpine regions, permafrost is seen as a stabilizing element [FKHN06, GH07, KFG13] and its retreat can lead to an increase in mass movements such as landslides and rockfalls [ACO11, RD11, FPH⁺12, BC20] threatening the (densely) populated mountainous regions [HRA⁺19]. Pertinent examples of mass movements are the rockfall and consequent debris flow which occurred 2017 at Piz Cengalo (CH) [WAK⁺20] or the 2021 rock and ice avalanche in Chamoli, Indian Himalaya [SJS⁺21], leading to a number of fatalities and severe damage to the built environment. The complex processes leading to such natural hazards are not thoroughly understood. Especially, the impact of climate change as trigger for such isolated hazardous events requires further research [SJS⁺21]. Nonetheless, high-mountain slopes are sensitive to warming from climate changes and an increase in frequency of such hazardous events is to be expected [HRA⁺19]. Therefore it is important to understand the variability and processes of climate change through monitoring, modeling and forecasting, adapt to the increased occurrence of such hazardous events using protective measures and promote and work towards mitigation efforts to reduce negative human influence on climate.

Environmental monitoring using wireless sensor networks provides opportunities to gain novel insights useful for modeling and forecasting [HTB⁺08]. Moreover, such systems can be designed to act as a protective measure in form of a hazard warning system [GTWX11]. However, collecting and analyzing environmental data is a labour intensive task. Developing, deploying and running an environmental monitoring system using wireless sensor networks requires cross-disciplinary knowledge (for example geophysical science, hardware and software engineering, communication technology, data science). In addition, information

and knowledge resources tend to be fragmented and isolated. Moreover, analysis methods and interfaces are often not compatible, but the analysis of a heterogeneous mixture of data streams requires well-aligned analytical tools with clearly defined interfaces.

Long-term analysis requires a careful consideration of short-term effects which consequently requires to analyze high-resolution short-term data for periods of up to many years; performing it manually is an unrealistic task. There are high expectations that advanced data science concepts and machine learning methods are capable to fill this gap. They have proven to be effective in various domains, such as image classification, audio event detection [TGPG16, HCE⁺16], debris flow detection [CWW⁺21] and rockfall detection [HPM⁺17, WHv⁺21]. However, when applied to long-term, real-world data, the effectiveness of these methods is often diminished by biases in training data [TSE⁺20], their requirement for careful post-processing [HPM⁺17] or susceptibility towards changes in data distribution, due to for example environmental changes, changes of sensor modalities, usage of different data capturing devices [RSG⁺21]. Most importantly, these methods usually require a clean, fully annotated dataset without label errors [NAM21].

In this dissertation, we focus on developing techniques to harness environmental monitoring data collected for the analysis of thawing permafrost as well as exploring technology for protective measures in form of rockfall early-warning. We acknowledge the intricate nature of real-world data by presenting methodologies which harmonize the interaction between experts of various disciplines as well as machine learning algorithms and tools to develop solutions for noisy, partially labeled and incomplete datasets.

It was of particular interest that parts of the resulting technologies may be used to raise awareness to consequences of negative human influence on climate. Consequently, we make data, code and publications public whenever possible. Moreover, we present a web-based data analysis and visualization tool for involving experts as well as the public.

1.1 Environmental monitoring strategies

Analyzing environmental data requires a careful design of the analysis system, including experimental considerations as well as technical possibilities. These design choices can be roughly categorized into the following four categories which will be further explained in the upcoming sections.

Context To which application scenario should the analysis be applied? The context is the base for design choices concerning the experimental setup and/or the monitoring system. In Section 1.2 we will define the context for this dissertation.

Data Basis Which data sources are required, used and how are they combined? The data basis may include sensors of different types, annotations or external data providers which can be used in various combinations which will be further discussed in Section 1.3.

Information Extraction Which techniques are applied to the data basis for information extraction and subsequent knowledge generation? Different types and amounts of data require different information extraction techniques. Two techniques will be highlighted in Section 1.4.

Computing Which is the optimal compute location in terms of capacity, throughput, latency, availability? Depending on the three aforementioned categories there are different requirements on computing which will be discussed in Section 1.5.

Overall, environmental monitoring can be regarded as an iterative process in which the above mentioned categories are repeatedly evaluated and refined. One such feedback loop is illustrated in Figure 4.5 which displays a wireless sensor network deployed with a certain set of sensors based on expert knowledge. The sensor data is transmitted for analysis, which is used for knowledge generation and refinement. This knowledge can be incorporated into a new or updated sensor deployment, yielding more relevant data for a more accurate analysis.

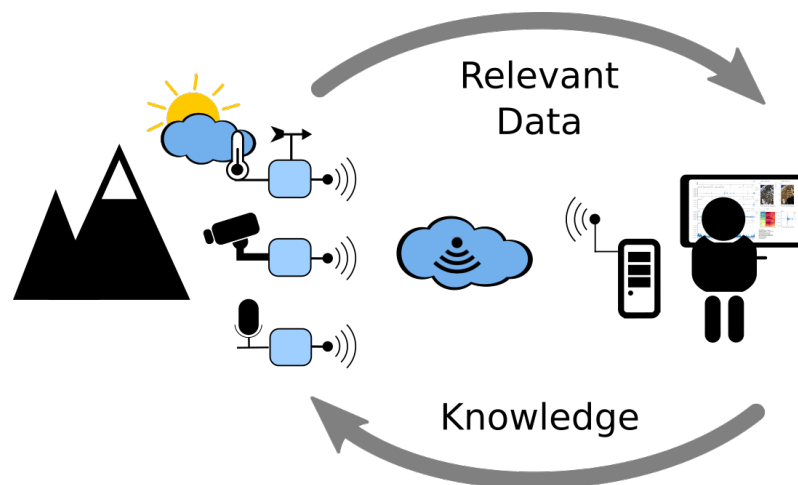


Figure 1.1 Iterative process of incorporating knowledge into a sensor network deployment to gather relevant data which in turn generates new knowledge.

1.2 Context: Applications of Wireless Sensor Networks

Wireless sensor networks are typically applied in scenarios which require sensor placement at locations which are hard to access or where networking

and power infrastructure is lacking. Several applications have emerged in various contexts. Wireless sensor networks were deployed for structural monitoring such as bridges [SHW⁺10] and tunnels [MPC⁺10, SHW⁺10] or railways [FGO⁺10]. They are frequently used for habitat monitoring [ZSLM04, BMC⁺11]. Moreover, environmental monitoring applications include volcano monitoring [WLR⁺06], air-quality monitoring [ASC⁺10] and permafrost monitoring [THGT07, HTB⁺08].

The advantages of wireless sensor networks are their independence from the power supply infrastructure, which is also one of their weak points. The reliance on battery power poses challenges in terms of sensor and network availability. These challenges can be met by using energy-efficient technology (microcontroller, memory, disk, battery), decoupling of communication and processing [SZDF⁺15, BTF⁺19], better battery technology, optimizing transmission protocols [FZTS11, FZMT12, SDFG⁺17] and using energy harvesting supported by a battery [DT21] or replacing the battery [GSS⁺17]. However, especially harsh environments such as steep high-alpine terrain pose an even stronger requirement on the battery life due to cold temperatures and the location's inaccessibility. Moreover, the impact of data loss due to transmission failures must be considered during experiment design which can only partially be reduced by using advanced communication protocols [FZTS11].

In this thesis, we focus on two specific applications of wireless sensor networks in which the above mentioned issues can be encountered, namely natural hazard warning and environmental science. Although the context is similar there are significant differences which affect the design choices of the data collection and analysis system.

Natural hazard warning requires a fast reaction time and fast decision making to avoid catastrophes. A warning system must be continuously running, have a high reliability (networking and power) and it must be very accurate during and before a hazard. Long-term data availability and thus a comprehensive dataset is usually not required but can be useful for system fine-tuning and later analysis.

Environmental science usually requires high-quality data for sound reasoning but not an ability to react fast. The focus is on general understanding and model building, which usually requires expert knowledge ideally supported by a long-term and comprehensive dataset. Here, it is not always required to monitor continuously if the time period of interest is known (short-term measurement campaigns). However, continuous monitoring can provide insights about the evolution of geophysical processes and can bring forward insights previously unnoticed.

Ideally, a monitoring system is suited to fulfill the requirements of both scenarios. One example could be a system continuously recording data which is capable of autonomously reacting to rare events or natural hazards by triggering an alarm and a high-resolution recording of the event. In practice such a system

is challenging to implement due to the constraints imposed by the choice of sensor and computing technology as well as information extraction methods. In this dissertation, we will try to find a good tradeoff given the constraints to develop systems applicable to one or both scenarios.

1.3 Data Basis: Matterhorn Deployment

Large parts of this dissertation are based on a long-term monitoring deployment at the Matterhorn Hörnligrat field site located in Zermatt, Switzerland at 3500 m a.s.l. [WBF⁺19]. Although other data is also used in this dissertation, such as seismic data from another deployment (Chapter 4) and acoustic data (Chapter 3 and 4), we specifically highlight the Matterhorn deployment since it is the focal point of this dissertation.

The Matterhorn field site was of special interest to conduct research on permafrost in steep bedrock as a result of a rockfall in summer 2003. Uncovered ice in the failure plain raised questions about the link between thawing permafrost and the rockfall. Located on a ridge, accessible by a frequently used climbing route, the location provided good preconditions to supplement previous studies on thermal behavior of steep permafrost bedrock [GKK⁺04, GHH04, NGK⁺07] with evidence from long-term, in situ measurements. Starting in 2007 with an initial sensor installations [THGT07, HTB⁺08], the deployment subsequently grew into a multi-sensor wireless sensor network [BGH⁺09a] using a heterogeneous set of sensors types, including thermal sensors, crackmeters, a high-resolution camera [KYB09], GNSS receivers [BBF⁺11b], local weather station, net total radiometer, acoustic emission sensors [GBG⁺12], accelerometer and seismometer [WFM⁺18]. Moreover, it has been used as a testbed and proof-of-concept for wireless communications protocols such as Dozer [BvW07] or the event-based Low-power Wireless Bus (eLWB) [SDFG⁺17] which is based on Glossy [FZTS11].

The data basis of this dissertation, is coming from a subset of the available sensors, namely rock temperature, weather station, radiometer, camera and seismometer. Additionally, we extend the sensor portfolio by a custom-designed, event-triggered seismic sensor.

In this dissertation, we will work with the selected data basis in two distinct ways.

Independent data analysis extracts information from each data type individually. For example by using photos to detect snow cover and seismic data to detect rockfalls.

Data fusion analysis extracts information jointly from a collection of data types. For example a machine learning model is trained jointly with photos and seismic data for mountaineer classification.

1.4 Information Extraction

In the upcoming chapters we will consider multiple applications where information extraction from the gathered signals is necessary to accomplish a specific task. In this dissertation, the task usually involves classifying an event which is registered by a sensor or a set of sensors. A concrete example is to detect a rockfall with a seismic sensor while making sure that the event registered is not wind or a mountaineer. We will discuss two options to extract the information about a rockfall occurrence from the gathered signals, namely by manual analysis or by algorithmic analysis. In our scenario, manual analysis requires a certain domain knowledge, which can be contributed by domain experts who work with the data using analytical tools and visualizations resulting in a model of the geophysical process. Relying solely on manual analysis is however not scalable to large datasets. Automating analysis by applying algorithms at scale has the potential to overcome this bottleneck. However, many algorithms cannot perform as well as a human expert. Machine learning is suited to integrate domain knowledge via expert-labeled datasets and it is applicable to large datasets. It has been shown in other application domains that machine learning algorithms come close to the accuracy of humans in classification tasks. However, these algorithms require a large training dataset of high quality which covers the full range of the expected data, it might introduce biases due to non-representative training sets, it is often susceptible to slight data modifications reducing its transferability and it is limited by available compute power.

Since a major part of this dissertation deals with the question of how to extract information from data, we will explain two options (involving domain knowledge or using machine learning) in more detail.

1.4.1 Domain Knowledge

Domain experts usually make use of a variety of analytical tools, experiments and datasets to develop a model explaining the (geophysical) processes under consideration. In the following, we will narrow our focus to one of such analytical tools, namely information extraction using preprocessing and visualization tools, since these tools can be used to condition the data basis using domain knowledge. As such, it can be the point of contact when joining machine learning and domain knowledge. Visually analyzing data is usually performed by viewing the signal in one or multiple representations obtained by preprocessing. For example, seismic data is usually analyzed using the waveform, the spectrogram, and/or the power spectral density [PMH⁺18]. In many cases, experts can identify a characteristic rockfall signature in the seismic signal [PMH⁺18], however deployment-dependent factors affect the recognition accuracy, such as the type of seismic instruments or anthropogenic noise source in proximity to the deployment. As a result, more information than just one sensor can help to identify the correct source of the event triggering

the seismometer. This information includes among others statistics, sensors of different types, eye-witness reports and expert annotations.

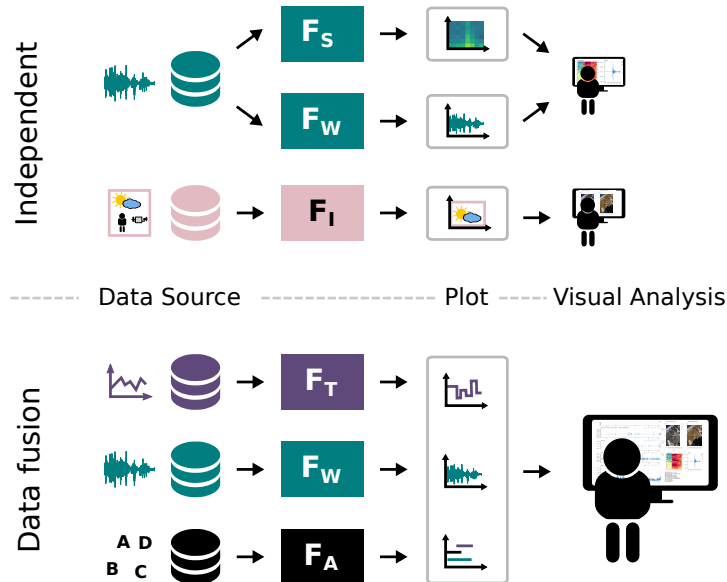


Figure 1.2 Expert-based information extraction can be categorized into analysis based on independent data or data fusion. For independent data, each data signal (here, **W**aveform and **I**mages) is fed independently through a **F**rontend conditioning the signal into a representation suitable to for visual analysis using a **P**lot. Finally, the generated plots are individually analyzed by data type. For data fusion, the data signals (here, **T**ime-series, **W**aveform, **A**nnotations) are fed through the frontend individually and are jointly plotted. The resulting plots are analyzed by an expert.

Figure 1.2 conceptually illustrates how to extract information by visual analysis from a set of signals. The available data is compiled based on their data type. Each data type may require individual transformations to extract useful information, for example spectrogram transformation, image resizing and cropping, time series aggregation. These transformations can also include models developed by domain experts. The data can be individually viewed (independent data) or placed side-by-side to be viewed jointly (data fusion) for example by viewing a segment of seismic data jointly with the timelapse image at the same time.

Navigating through data streams of large and heterogenous datasets is inherently complex in terms of data processing, visualization and information extraction. Therefore, a method to manage a huge amount of data or to reduce the amount of information to a manageable subset is favorable. In this dissertation we will work on algorithms to preselect informative subsets of a dataset which can be subsequently analyzed. Moreover, we present a framework to manage multiple data sources for joint processing, visualization and integration of algorithms into the analysis workflow, such as predictions obtained via machine learning algorithms.

1.4.2 Machine Learning

Machine learning in general, and deep learning in particular, have gained a lot of interest due to their advanced feature extraction and classification capabilities. In contrast to manual model development using domain knowledge, machine learning algorithms are used to automatically develop a model based on data. The advantage of machine learning is the fact that models can be developed even if the underlying processes to be modeled are not well understood or domain knowledge is missing. These models can surpass human capabilities in certain tasks. However, they usually require a large amount of annotated data and in certain cases the lack of expert knowledge leads to erroneous models. In this dissertation we will focus on a subset of deep learning, namely artificial neural networks such as multilayer perceptrons (MLP) and convolutional neural networks (CNN).

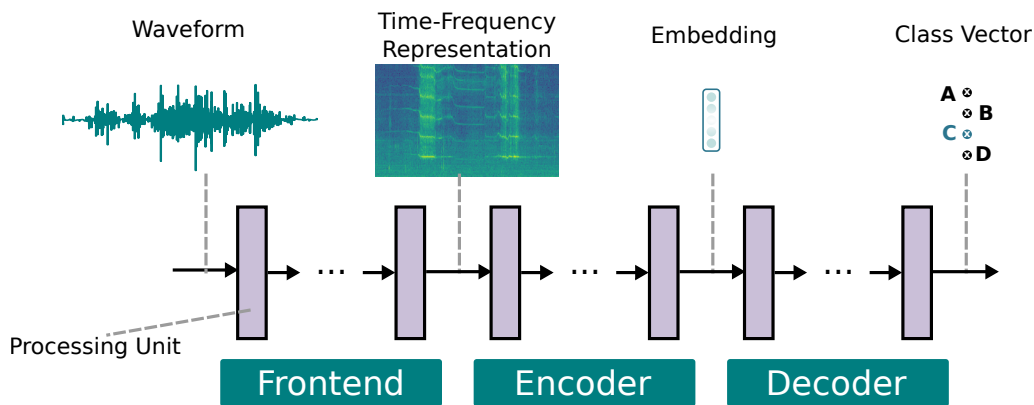


Figure 1.3 High level overview of the processing pipeline used throughout this dissertation.

On a high level, the processing pipeline used in this dissertation can be subdivided into the three parts, namely frontend, encoder and decoder as illustrated in Figure 1.3 for the example of waveform classification. The input data is preprocessed using one or many processing units (for example detrending the signal, applying the short-time Fourier transform). The result is fed into an encoder which transforms the data into an intermediate representation (embedding) which is then further transformed into a meaningful output (class vector) using a decoder. The encoder and decoder usually consist of processing units with adaptable parameters (for example layers of an artificial neural network). These parameters are optimized in an iterative, data-driven procedure to make the output meaningful. This optimization procedure makes the processing pipeline generically applicable to a large range of data input types and a large range of different tasks. Figure 1.3 also illustrates that the data-driven optimization of machine learning creates a "black box" model. In contrast to the waveform, spectrogram or class names, the embedding produced by the encoder is usually a high-dimensional vector or matrix which is typically

non-intuitive and meaningless to the human eye.

In analogy to Figure 1.2, Figure 1.4 conceptually illustrates how machine learning models can be optimized. The available data is compiled based on their data type and each data item is fed through the frontend and encoder to compute the embedding. The embedding usage depends on the use case. For optimization based on independent data, the embeddings are independently fed through a decoder and the combination of encoder and decoder are optimized using a target-based optimization, such as computing a classification loss (L_C) between decoder output and annotations or optimizing to recreate the input signal (L_R). For optimization based on data fusion, we distinguish two cases, namely target-based optimization or embedding-based optimization. These two optimizations can be applied simultaneously or sequentially. Target-based optimization is applied the same way as for optimization of independent data

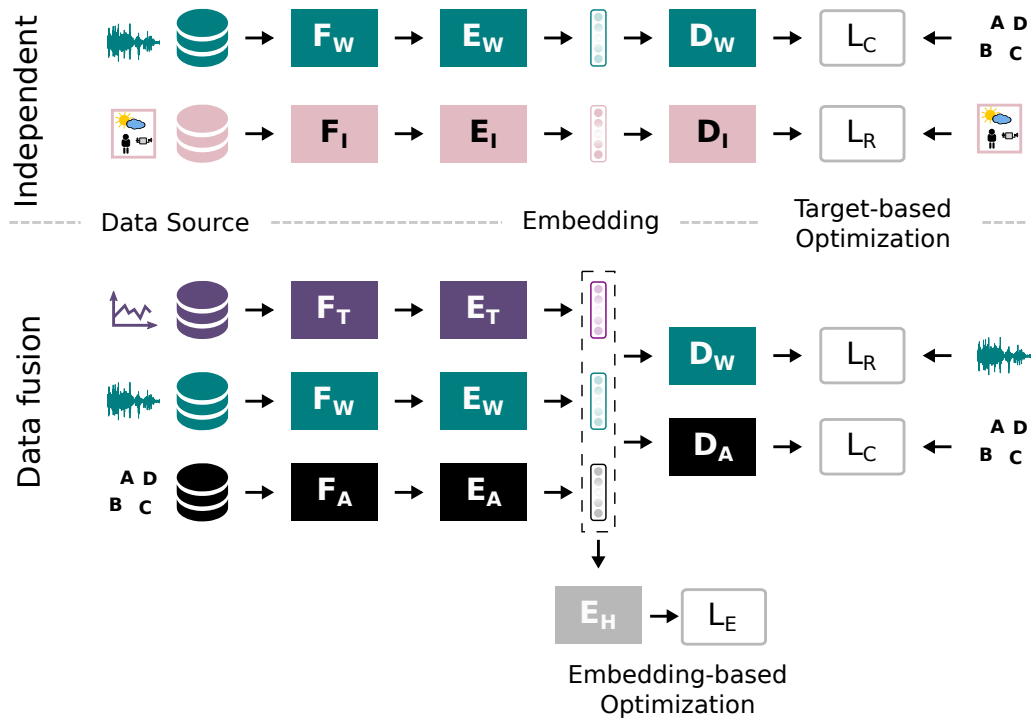


Figure 1.4 Machine learning-based information extraction can be categorized into optimization based on independent data or data fusion. For independent data, each data signal (here, **Waveform** and **Images**) is fed independently through a chain of processing units consisting of a **Frontend**, **Encoder** and **Decoder**. Finally, the processing units which are trainable are optimized using a loss function (**Classifier Loss** L_C) and corresponding ground truth. For data fusion, the data signals (here, **Time-series**, **Waveform**, **Annotations**) are fed through the chain of frontend and encoder independently. The resulting embedding of each chain can be fed through a decoder being optimized with target-based optimization (classification loss L_C or recreation loss L_R) or fed through an encoder head E_H being optimized using an embedding loss L_E .

but may be designed to optimize all encoders simultaneously. For embedding-based optimization the embedding is fed through an encoder head and the result is used to compute an embedding loss which in turn updates the encoders and encoder head.

In the upcoming chapters we will define concrete implementations of these optimization procedures. Note that the division into components as presented above is purely conceptual. In application, there is often no clear division. For example in most classification tasks, encoder and decoder are optimized end-to-end and represented by one single artificial neural network. However, in Chapter 3 and in Chapter 4 we will use this distinction to optimize a model architecture and develop a semi-supervised method, respectively.

In this dissertation, we will apply machine learning methods to classification problems in environmental monitoring. We will design methods to integrate with but not replace the prevailing analysis methods. Especially, we design methods to clean monitoring data of unwanted influences in post-processing and during data recording. Moreover, we introduce options to combine domain knowledge and machine learning by using expert-based annotations and domain knowledge about the system context. Finally, machine learning is made accessible with a framework to process, visualize and annotate data as well as processing results.

1.5 Computing

The question where to compute is important and depends on the requirements highlighted in the previous sections, namely on the amount of data to be processed, what type of processing to perform and most importantly on the context. Computation is usually limited by capacity (how much can be stored?), throughput (how much/fast can be processed?) and latency (how long does it take to get the first results for a query?). Especially in wireless sensor networks, availability of a service is another significant factor (how steady is the energy supply? How often is a sensor non-responsive?). Computation can be roughly subdivided into edge computing (close to the sensor on an embedded device) or cloud computing (on a server in a datacenter).

Edge computing provides fast response times (low latency) but is limited by energy (availability), processing speed (throughput) and memory/storage (capacity). Examples for edge computing are on-sensor data preprocessing or more complex task such as on-device keyword spotting [ZSLC17] using machine learning.

Cloud computing is an umbrella term for storage and applications running on dedicated network-accessible compute hardware. Here, compute and storage resources are virtually unlimited (high throughput and capacity). Availability is limited mainly by the connection from server to user. Server-sided availability is usually guaranteed by the cloud vendors (high availability). However, in contrast

to computing on the edge, transferring data from "the edge into the cloud" introduces an additional delay (high latency) and additional resource usage such as communication, bandwidth and energy. Depending on the application, this delay is non-negligible.

In this dissertation we will try to balance the tradeoff between accurate but computationally complex deep learning models and resource limited computation devices. On the one hand, we focus on architectural and computational optimizations which allow a deep learning model to run on resource-constraint edge devices. On the other hand, we are developing cloud-based analytics tools that leverage the high availability and computing power of the cloud to analyze long-term datasets collaboratively with multiple people.

1.6 Thesis Outline and Contributions

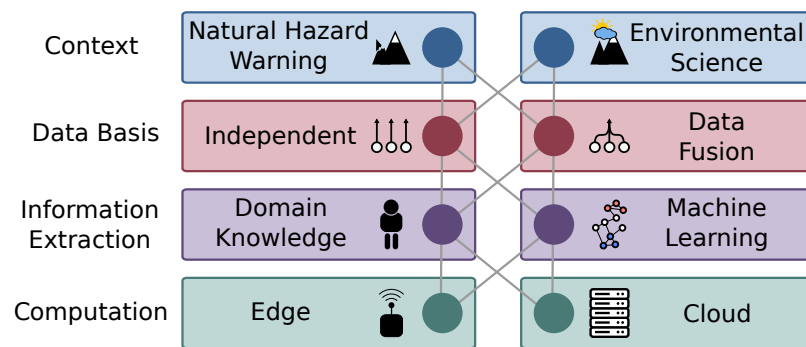
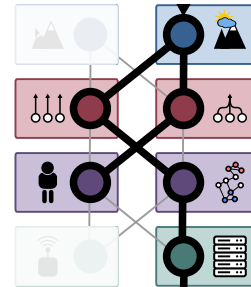


Figure 1.5 Conceptual structure of this dissertation. Each of the main categories is defined by a concept pair - two conceptually opposing aspects of each category.

Summarizing the previous sections, Figure 1.5 highlights the main design criteria for a system analyzing environmental data. Each of the previously discussed categories (namely context, data basis, information extraction and computation) is defined by two conceptually opposing aspects: The context is defined by applications to natural hazard warning and/or environmental science. The data basis can be based on independent data and/or data fusion. The information extraction can be done using domain knowledge and/or machine learning. The computation can be performed on the edge and/or in the cloud. In the following we will present the contributions of each chapter and highlight the related aspects.

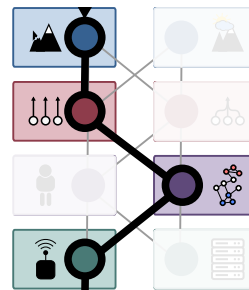
Chapter 2

In Chapter 2, we first introduce a specific set of sensors of the wireless sensor network deployment at Matterhorn, Switzerland and its characteristics. Using data from this set of sensors, we propose a methodology to systematically filter out unwanted influences from seismic signals using a set of heterogenous sensors and human-in-the-loop.



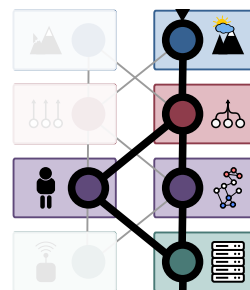
Chapter 3

In Chapter 3, we present an event-driven, low-power system for the purpose of rockfall early-warning. Additionally, the system features a method for timely, on-device processing of a convolutional neural network. It is deployed in the aforementioned Matterhorn field site and its characteristics are evaluated.



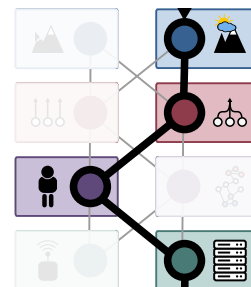
Chapter 4

In Chapter 4, on the one hand we demonstrate how machine learning can support domain-experts with semi-supervised and unsupervised information extraction. On the other hand, we demonstrate how experts increase performance of machine learning classifiers by providing crucial knowledge about the system context, collecting such knowledge in form of a graph and utilizing the graph in a deep representation learning framework.



Chapter 5

Finally, in Chapter 5, we present the data analytics and visualization platform *foReal* which allows to combine environmental data from different sources, enabling fast and easy exchange between experts of various disciplines while fulfilling the processing demands of long-term, large-scale environmental datasets.



2

Identification of external influences in micro-seismic recordings

In this chapter, we introduce the general setting of this thesis, namely using seismic monitoring and wireless sensor networks for long-term monitoring of high-alpine steep permafrost bedrock. Furthermore, we propose a methodology to systematically filter out unwanted influences from seismic signals using a set of heterogeneous sensors and human-in-the-loop.

Passive monitoring of ground motion can be used for geophysical process analysis and natural hazard assessment. Detecting events in micro-seismic signals can provide responsive insights into active geophysical processes. Novel sensor network technology and cheaper seismic sensors are facilitating applications such as local detection of mass movements and slope instabilities allowing for a quick deployment in hard to reach regions. However, in the raw signals micro-seismic events are superimposed by external influences, for example anthropogenic or natural noise sources that distort analysis results. In order to be able to perform event-based, geophysical analysis with such micro-seismic data records it is imperative that negative influence factors can be systematically and efficiently identified, quantified and taken into account. Current identification methods (manual and automatic) are subject to variable quality, inconsistencies or human errors. Moreover, manual methods suffer from their inability to scale to increasing data volumes, an important property when dealing with very large data volumes as in the case of long-term monitoring.

In this chapter, we present a systematic strategy to identify a multitude of external influence sources, characterize and quantify their impact and develop methods for automated identification in micro-seismic signals. We apply the strategy developed to a real-world, multi-sensor, multi-year micro-seismic monitoring experiment performed at the Matterhorn Hörnligrat (CH). We develop and present an approach based on convolutional neural networks for micro-seismic data to detect external influences originating in mountaineers, a

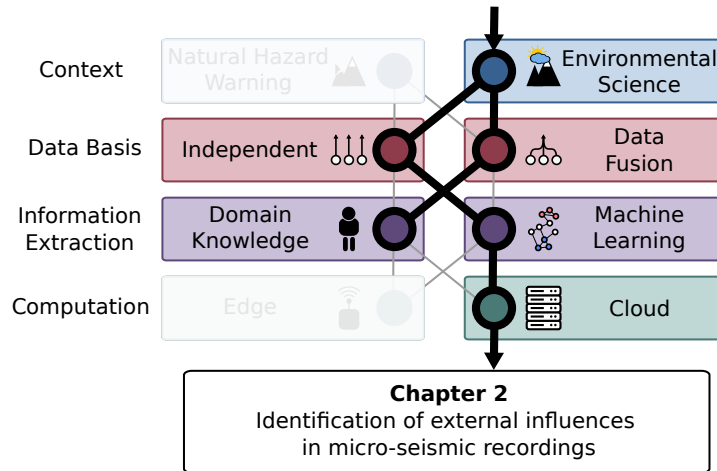


Figure 2.1 This chapter's position in the context of the dissertation

major unwanted influence, with an error rate of less than 1%, 3x lower than comparable algorithms. Moreover, we present an ensemble classifier for the same task obtaining an error rate of 0.79% and an F1 score of 0.9383 by jointly using time-lapse image and micro-seismic data on a annotated subset of the monitoring data. Applying these classifiers to the whole experimental dataset reveals that approximately 1/4 of events detected by an event detector without such a pre-processing step are not due to seismic activity but due to anthropogenic influences and that time periods with mountaineer activity have a 9x higher event rate. Due to these findings we argue that a systematic identification of external influences using a semi-automated approach and machine learning techniques as presented in this paper is a prerequisite for the qualitative and quantitative analysis of long-term monitoring experiments.

2.1 Problem Setting

Passive monitoring of elastic waves, generated by the rapid release of energy within a material [Har03] is a non-destructive analysis technique allowing a wide range of applications in material sciences [LCC01], engineering [GO08] and natural hazard mitigation [MCO12] with recently increasing interest into investigations of various processes in rock slopes [AAC+10, OCA+12]. Passive monitoring techniques may be broadly divided into three categories, characterized by the number of stations (single vs. array), the duration of recording (snapshot vs. monitoring) and the type of analysis (continuous vs. event-based). On the one hand, continuous methods such as the analysis of ambient seismic vibrations can provide information on internal structure of a rock slope [BMMF12, GEMH15, WFB+18]. On the other hand, event-based methods such as the detection of micro-seismic events (which are focus of this study) can give immediate insight into active processes, such as local irreversible (non-elastic) deformation occurring due to the mechanical loading of rocks [GO08]. However, for the reliable detection of events irrespective of the detection method the signal source of concern has to be distinguishable from noise, for example background seismicity or other source types. This discrimination is a common and major problem for analyzing micro-seismic data.

In general, event-based geoscientific investigations focus on events originating from geophysical sources such as mechanical damage, rupture or fracture in soil, rock and/or ice. These sources originate for example in thermal stresses, pressure variations or earthquakes [AGG12]. However, non-geophysical sources can trigger events as well: (i) anthropogenic influences such as helicopter or mountaineers [ELBA17, vS11, WFM+18] and (ii) environmental influences / disturbances, such as wind or rain [AAC+10]. One way to account for such external influences is to manually identify their sources in the recordings [vS11]. This procedure, however, is not feasible for autonomous monitoring because manual identification does not scale well for increasing amounts of data. Another approach is to limit to field sites far away from possible sources of uncontrolled (man-made) interference or to focus and limit analysis to decisively chosen time periods known not to be influenced by for example anthropogenic noise [OCA+12]. In practice both the temporal limitation as well as the spatial limitation pose severe restrictions. First, research applications can benefit from close proximity to man-made infrastructure since set up and maintenance of monitoring infrastructure is facilitated [WLJ+06]. Second, applications in natural hazard early warning must not be restricted to special time-periods only. Moreover, they are specifically required to be usable close to inhabited areas since those should be protected, although this setting increases the likelihood for human interference on the signals recorded. As a conclusion it is a requirement that external influences can be taken into account with an automated workflow, including pre-processing, cleaning and analysis of micro-seismic data.

A frequently used example of an event detection mechanism is an event detector

called STA/LTA that bases on the ratio of short-term average to long-term average [All78]. Due to its simplicity, this event detector is commonly used to assess seismic activity by calculating the number of triggering events per time interval for a time period of interest [WAY+98, AGS05, SDL09]. It is often used in the analysis of unstable slopes [CCVB18, LJB11] and is available integrated into many commercially available digitizers and data loggers [Geo18]. With respect to unwanted signal components, STA/LTA has also been used to detect external influence factors such as footsteps [AMK18] but due to its inherent simplicity, it cannot reliably discriminate geophysical seismic activity from external (unwanted) influence factors such as noise from humans, natural sources like wind, rain or hail without manually supervising and intervening with the detection process on a case by case basis. As a result the blind application of STA/LTA will inevitably lead to the false estimation of relevant geophysical processes if significant external influences, such as wind, are present [All78].

There exist several algorithmic approaches to mitigate the problem of external influences by increasing the selectivity of event detection. Unsupervised algorithms such as auto-correlation [BBS08, AB14, YOBB15] are either computationally complex or do not perform well for low signal to noise ratios. Supervised methods can find events in signals with low signal to noise ratio. For example template matching approaches such as cross-correlation methods [GR06] use event examples to find similar events, failing if events differ significantly in "shape" or if the transmission medium is very inhomogeneous [WFM+18]. The most recent supervised methods are based on machine learning techniques [RA17, OCB18] including the use of neural networks [KG17, PGD18, LMH+18, RMH18]. These learning approaches show promising results with the drawback that large datasets containing ground truth (verified events) are required to train these automated classifiers. In earthquake research large databases of known events exist [KASK16, RMH18], but in scenarios like slope instability analysis where effects are on a local scale and specific to a given field site such data are inexistent. Here, inhomogeneities are present on a very small scale and field sites differ in their specific characteristics with respect to signal attenuation and impulse response. In order to apply such automated learning methods to these scenarios obtaining a dataset of known events is required for each new field site requiring substantial expert knowledge for a very arduous, time-consuming task. The aim of this study is to use a semi-automatic workflow to train a classifier which enables the automatic identification of unwanted external influences in real-world micro-seismic data. By this means, the geophysical phenomena of interest can be analyzed without the distortions of external influences.

To address these problems, this chapter contains the following contributions.

- We propose a strategy to identify and deal with unwanted external influences in multi-sensor, multi-year experiments.
- We compare the suitability of multiple algorithms for mountaineer detection using a combination of micro-seismic signals and time-lapse images.

- We propose a convolutional neural network (CNN) for source identification.
- We exemplify our strategy for the case of source identification on real-world micro-seismic data using monitoring data in steep, fractured bedrock permafrost.
- We further provide the real-world micro-seismic and image data as an annotated dataset containing data from a period of two years as well as an open source implementation of the algorithms presented.

2.2 Methodology

In this chapter we present a systematic and automated approach to identify unwanted external influences in long-term, real-world micro-seismic datasets and preparing this data for subsequent analysis using a domain-specific analysis method, as illustrated in Fig. 2.2. Traditionally, the signal, consisting of the phenomena of interest and superimposed external influences, is analyzed directly. However, this analysis might suffer from distortion through the external influences. By using additional sensors like weather stations, cameras or microphones and external knowledge such as helicopter flight plans or mountain hut occupancy it is possible to semi-automatically label events originating from non-geophysical sources, such as helicopters, footsteps or wind without the need of expert knowledge. Such "external" information sources can be used to train an algorithm that is then able to identify unwanted external influence. Using this approach multiple external influences are first classified and labeled in an automated pre-processing step with the help of state-of-the-art machine learning methods. Subsequent to this classification, the additional information can be used for domain-specific analysis such as separating geophysical and unwanted events triggered by a simple event detector (for example an STA/LTA event detector). Alternatively, more complex approaches can be used taking into account signal content, event-detections and classifier labels of the external influences. However the specifics of such advanced domain-specific analysis methods is beyond the scope of this chapter and subject to future work. A basic example of a custom domain-specific analysis method is the estimation of separate STA/LTA event rates for time periods when mountaineers are present and when they are not which we use as a case study in the evaluation section of this chapter to exemplify our method.

Figure 2.3 illustrates the overall methodology in detail. In a first step the available data sources of a case study are assessed and cataloged. Given a case study (Sect. 2.3) consisting of multiple sensors, one or more sensor signals are specified as primary signals (for example the micro-seismic signals, highlighted with a light green arrow in the figure) targeted by a subsequent domain-specific analysis method. Additionally, secondary data (highlighted with blue arrows) are chosen to support the classification of external influences contained in the primary signal. Conceptually these secondary data can be of different nature,

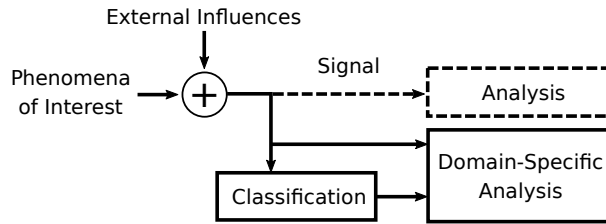


Figure 2.2 Real world measurement signals contain the phenomena of interest superimposed with external influences. If directly analyzed (dashed lines) the results are perturbed by the external influences. In contrast to this approach, in this chapter we suggest a systematic and automated approach to first identify a multitude of external influence sources in micro-seismic signals using a classifier. The classifier result data can then be used to quantify unwanted signal components as well as drive more extensive and powerful event detection and characterization methods leveraging combinations of both the signals as well labelled and classified noise data (solid lines).

either different sensor signals, e.g. time-lapse images or weather data or auxiliary data such as local observations or helicopter flight data. All data sources are combined into a dataset. However, this resulting dataset is not yet annotated as required to perform domain-specific analysis leveraging the identified and quantified external influences.

Two key challenges need to be addressed in order to establish such an annotated dataset by automatic classification: (i) suitable data types need to be selected for classification since not every data type can be used to continuously classify every external influence (for example wind sensors are not designed to capture the sound of footsteps; flight data may not be available for each time step) and (ii) a single (preferred) or at least a set of suitable, well-performing classifiers have to be found for each type of external influence source. Once these challenges have been solved a subset of the dataset is manually annotated in order to select and train the classifier(s) in a "preparatory" phase required to be performed only once, which includes manual data assessment (Sect. 2.4) as well as classifier selection and training (Sect. 2.5). The trained classifier is then used in an automated setup to annotate the whole dataset (Sect. 2.6). This "execution" phase can be performed in a one-shot fashion (post-processing all data in one effort) or executed regularly, for example on a daily or weekly basis if applied to continuously retrieved real-time monitoring data. These additional information can be used to perform a subsequent domain-specific analysis. This study concludes with an evaluation (Sect. 2.7) and discussion (Sect. 2.8) of the presented method.

2.3 Case Study

The data used in this chapter originate from a multi-sensor and multi-year experiment [WFM⁺18] focusing on slope stability in high-alpine permafrost

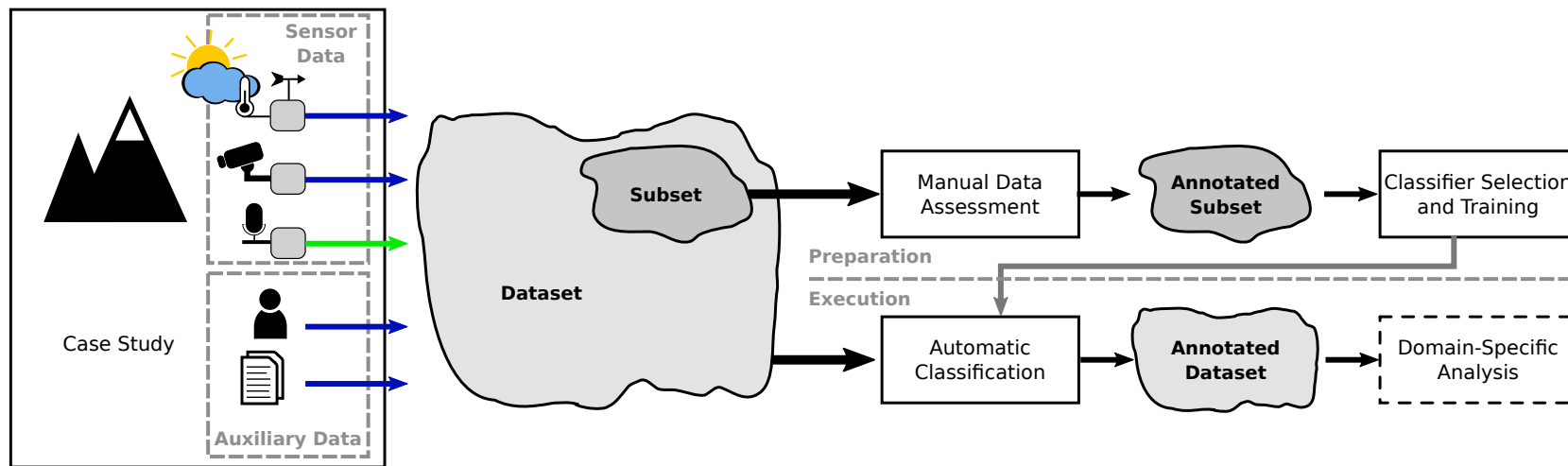


Figure 2.3 Conceptual illustration of the classification method to enable domain-specific analysis of a primary sensor signal (in our case micro-seismic signals denoted by the light green arrow) based on annotated datasets: A subset of the dataset containing both sensor and auxiliary data, is used to select and train a classifier that is subsequently applied to the whole dataset. By automatically and systematically annotating the whole dataset of the primary signal of concern, advanced methods can be applied that are able to leverage both multi sensor data as well as annotation information.

rock walls and understanding the underlying processes. Specifically, the sensor data is collected at the site of the 2003 rockfall event on the Matterhorn Hörnligrat, (Zermatt, Switzerland) at 3500 m a.s.l. where an ensemble of different sensors has monitored the rockfall scar and surrounding environment over the past ten years. Relevant for this chapter are data from a three-component seismometer (Lennartz LE-3Dlite MkIII), images from a remote controlled high-resolution camera (Nikon D300, 24 mm fixed focus), rock surface temperature measurements, net radiation measurements and ambient weather conditions, specifically wind speed from a co-located local weather station (Vaisala WXT520).

The seismometer applied in the case study presented is used to assess the seismic activity by using an STA/LTA event detector, which means for our application that the seismometer is chosen as the primary sensor and STA/LTA triggering is used as the reference method. Seismic data is recorded locally using a Nanometrics Centaur digitizer and transferred daily by means of directional WiFi. The data is processed on-demand using STA/LTA triggering. The high-resolution camera's [KYB09] field of view covers the immediate surroundings of the seismic sensor location as well as some backdrop areas further away on the mountain ridge. Figure 2.4 shows an overview of the field site including the location of the seismometer and an example image acquired with the camera. The standard image size is 1424x2144 pixels captured every 4 minutes. The Vaisala WXT520 weather data as well as the rock surface temperature are transmitted using a custom wireless sensor network infrastructure. A new measurement is performed on the sensors every 2 minutes and transmitted to the base station, resulting in a sampling rate of 30 samples per hour.

Significant data gaps are prevented by using solar panels, durable batteries and field-tested sensors but given the circumstances on such a demanding high-alpine field site certain outages of single sensors, for example due to power failures or during maintenance could not be prevented. Nevertheless this dataset constitutes an extensive and close-to-complete dataset.

The recordings of the case study were affected by external influences, especially mountaineers and wind. This reduced the set of possible analysis tools. Auxiliary data which helps to characterize the external influences is collected in addition to the continuous data from the sensors. In the presented case the auxiliary data is non-continuous and consist of local observations, pre-processed STA/LTA triggers from a previous study [WFM⁺18], accommodation occupancy of a nearby hut and a non-exhaustive list of helicopter flight data from a duration of approximately 7 weeks provided by a local helicopter company.

In following, we use this case study to exemplify our method presented in the previous sections.

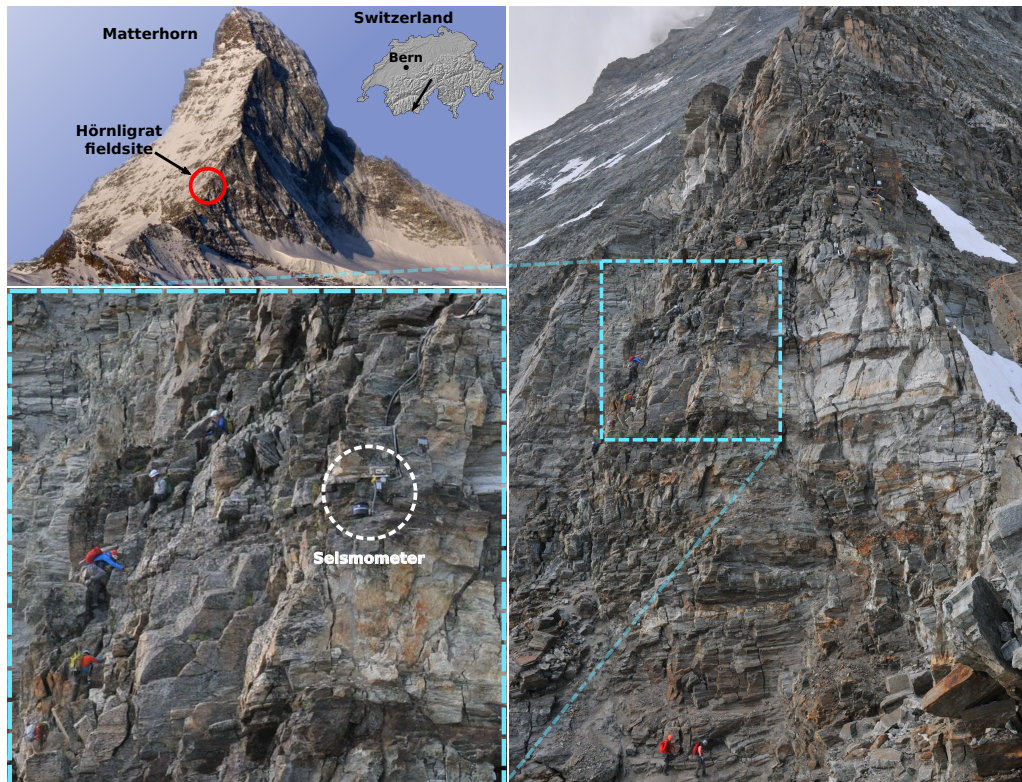


Figure 2.4 The field site is located on the Matterhorn Hörnligrat at 3500 m a.s.l. which is denoted with a red circle. The photograph on the right is taken by a remotely controlled DSLR camera on the field site at 2016-08-04T12:00:11. The seismometer of interest (white circle) is located on a rock instability which is close to a frequently used climbing route.

2.4 Manual Data Assessment

A ground truth is often needed for state-of-the-art classifiers (such as artificial neural networks). To establish this ground truth while reducing the amount of manual labor only a subset of the dataset is selected and used in a manual data assessment phase, which consists of data evaluation, classifier training and classifier selection as depicted in Fig. 2.5. Data evaluation can be subdivided into four parts: (i) characterization of external influences in the primary signal (that is the relation between primary and secondary signals), (ii) annotating the subset based on the primary and secondary signals, (iii) selecting the data types suitable for classification and (iv) performing a first statistical evaluation with the annotated dataset, which facilitates the selection of a classifier. Characterization and statistical evaluation are the only steps where domain expertise is required while it is not required for the time and labor intensive annotation process.

The classifier selection and training phase presumes the availability of a variety of classifiers for different input data types, for example the broad range of available image classifiers [RDS⁺15]. The classifiers do not perform equally well on the

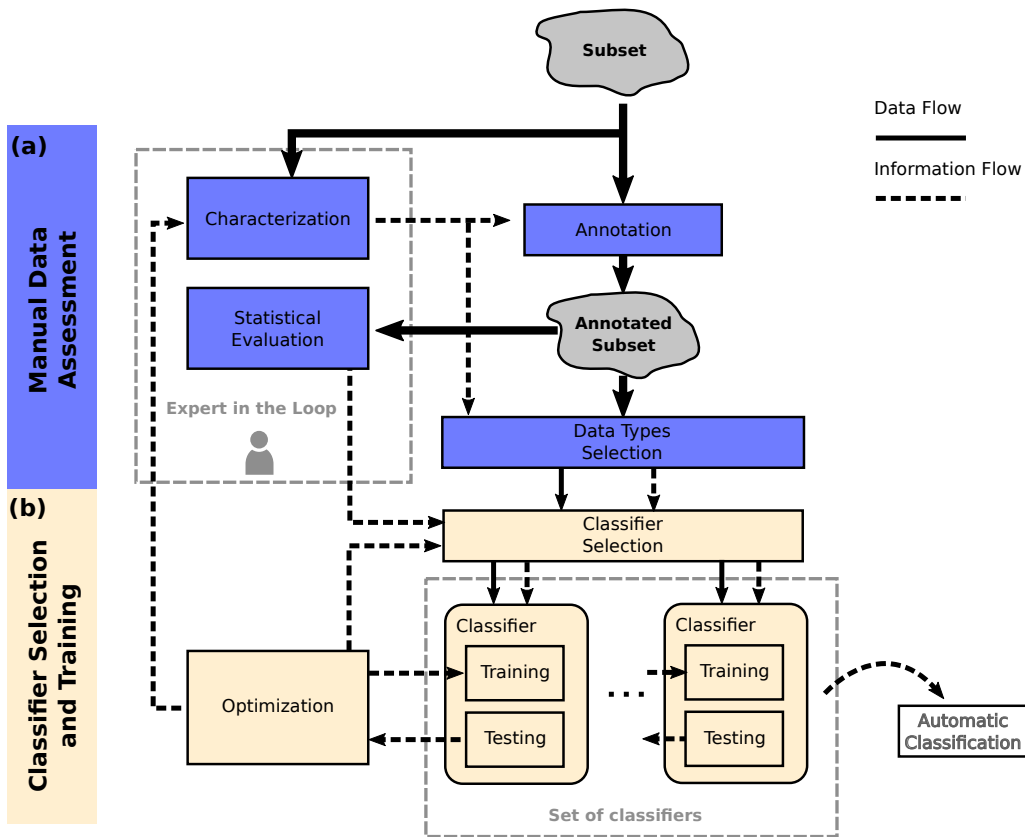


Figure 2.5 The manual preparation phase is subdivided into data evaluation (a) and classifier selection and training (b). First, the data subset is characterized and annotated. This information can be used to do a statistical evaluation and select data types which are useful for classification. Domain experts are not required for the labor intensive task of annotation. The classifiers are selected, trained and optimized in a feedback loop until the best set of classifiers is found.

given task with the given subset. Therefore classifiers have to be selected based on their suitability for classification given the task and the data. A selection of classifiers is therefore trained and tested with the annotated subset and optimized for best performance which can for example be done by selecting the classifier with the lowest error rate on a defined test set. The classifier selection, training and optimization is repeated until a sufficiently good set of classifiers has been found. This suitability is defined by the user and can for example mean that the classifier is better than a critical error rate. These classifiers can then be used for application in the automatic classification process.

In the following, the previously explained method will be exemplified for wind and mountaineer detection using micro-seismic, wind and image data from a real-world experiment. The required steps of subset creation, characterization, annotation, statistical evaluation and the selection of the data type for classification are explained. Before an annotated subset can be created the collected data must be characterized for its usefulness in the annotations

process, i.e. which data type can be used to annotate which external influence.

2.4.1 Characterization

The seismometer captures elastic waves originating from different sources. In this study we will consider multiple non-geophysical sources, which are mountaineers, helicopters, wind and rockfalls. Time periods where the before mentioned sources can not be identified are considered as relevant and thus we will include them in our analysis as a fifth source, the "unknown" source. The mountaineer impact will be characterized on a long-term scale by correlating with hut accommodation occupancy (see Fig. 2.11) and on a short-scale by person identification on images. Helicopter examples are identified by using flight data and local observations. By analyzing spectrograms one can get an intuition what mountaineers or helicopters "look" like, which facilitates the manual annotation process. In Fig. 2.6 different recordings from the field site are illustrated, which have been picked by using the additional information described at the beginning of this section. For six different examples the time domain signal, its corresponding spectrogram and STA/LTA triggers are depicted. The settings for the detector are the same for all the subsequent plots. It becomes apparent in Fig. 2.6 (b)-(c) that anthropogenic noise, such as mountaineers walking by or helicopters, are recorded by seismometers. Moreover, it becomes apparent that it might be feasible to assess non-geophysical sources on a larger time frame. Mountaineers for example show characteristic patterns of increasing or decreasing loudness and helicopters have distinct spectral patterns, which could be beneficial to classify these sources. Additionally, the images captured on site show when a mountaineer is present (see Fig. 2.4), but due to fog, lens flares or snow on the lens the visibility can be limited. The limited visibility needs to be taken into account for when seismic data is to be annotated with the help of images. Another limiting factor is the temporal resolution of one image every 4 minutes. Mountaineers could move through the visible area in between two images.

The wind sensor can directly be used to identify the impact on the seismic sensor. Figure 2.7 illustrates the correlation between tremor amplitude and wind speed. Tremor amplitude is the frequency-selective, median, absolute ground velocity and has been calculated for the frequency range of 1-60 Hz according to [BAW⁺15]. By manually analyzing the correlation between tremor amplitude and wind speed it can be deduced that wind speeds above approximately 30 km/h have a visible influence on the tremor amplitude.

Rockfalls can best be identified by local observations since the camera captures only a small fraction of the receptive range of the seismometers. Figure 2.6 (e) illustrates the seismic signature of a rockfall. The number of rockfall observations and rockfalls caught on camera are however very limited. Therefore it is most likely that we were unable to annotate all rockfall occurrences. As a consequence we will not consider a rockfall classifier in this study.

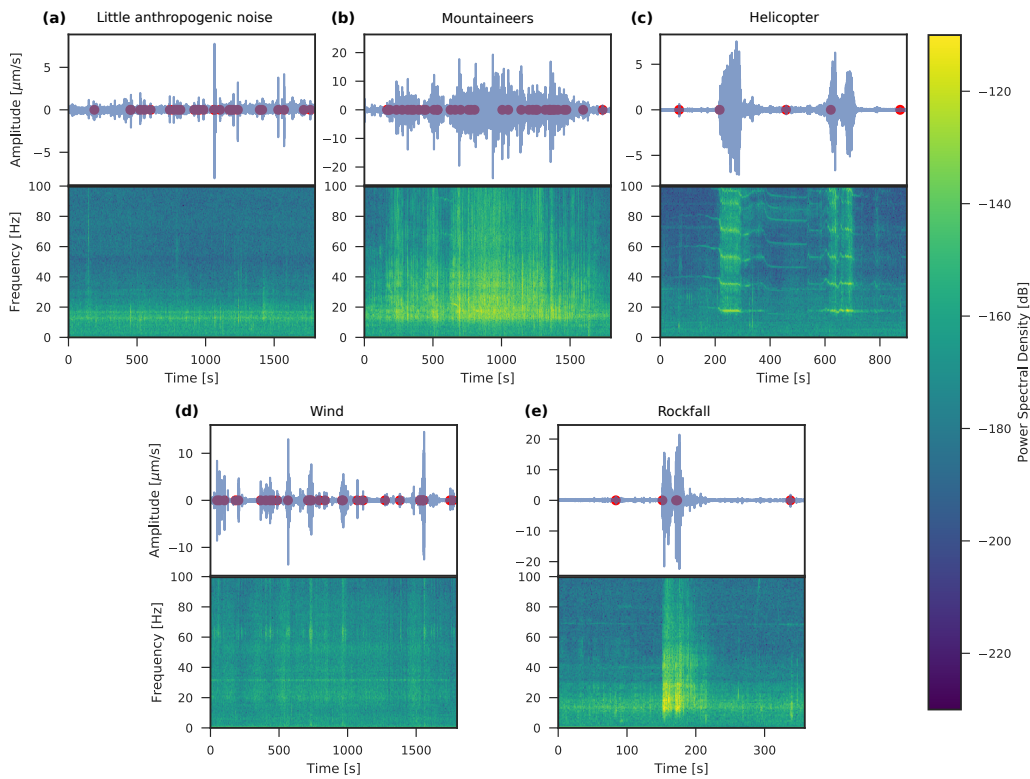


Figure 2.6 Micro-seismic signals and the impact of external influences: (a) During a period of little anthropogenic noise the seismic activity is dominant. (b) In the spectrogram the influence of mountaineers become apparent. Shown are seismic signatures of (c) a helicopter in close spatial proximity to the seismometer (d) wind influences on the signal (e) a rockfall in close proximity to the seismometer. The red dots in the signal plots indicate the timestamps of the STA/LTA triggers from [WFM⁺18].

It can be seen in Fig. 2.6 (a) that during a period which is not strongly influenced by external influences the spectrogram shows mainly energy in the lower frequencies with occasional broadband impulses.

The red circles in the subplots in Fig. 2.6 indicate the timestamps of the STA/LTA events for a specific geophysical analysis [WFM⁺18]. By varying the threshold of the STA/LTA event trigger the number of events triggered by mountaineers can be reduced. However, since the STA/LTA event detector cannot discriminate between events from geophysical sources and events from mountaineers, changing the threshold would also influence the detection of events from geophysical sources. This fact would affect the quality of the analysis since the STA/LTA settings are determined by the geophysical application [CCVB18, WFM⁺18].

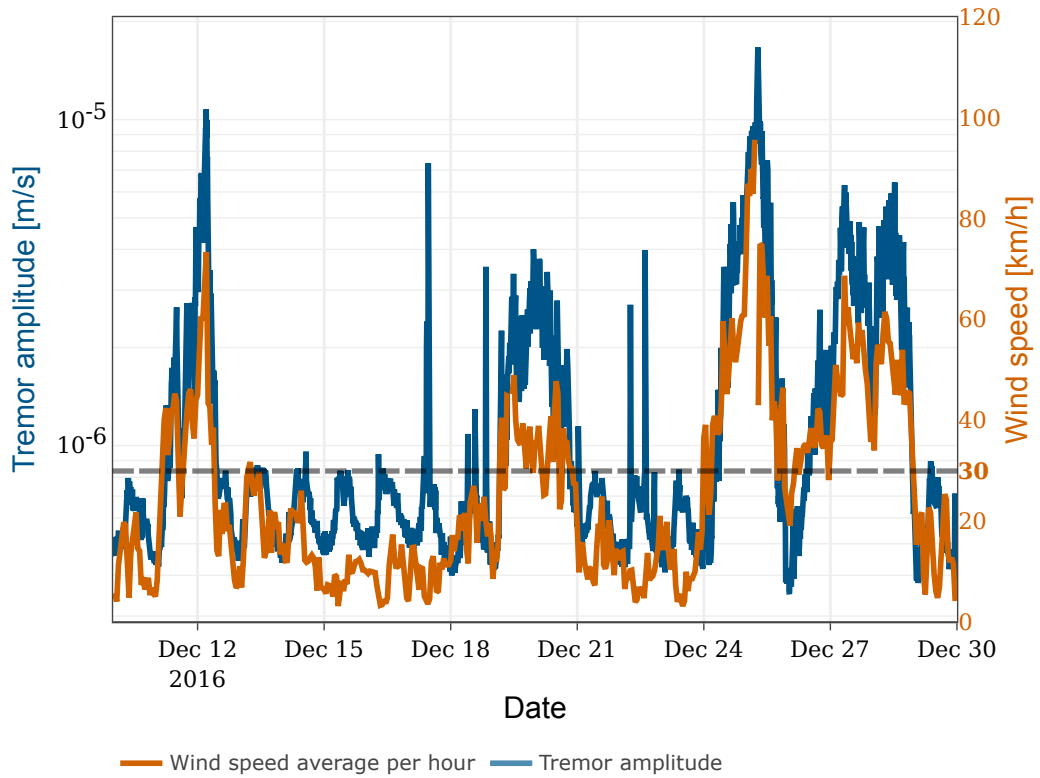


Figure 2.7 Impact of wind (light orange) on the seismic signal. The tremor amplitude (dark blue) is calculated according to [BAW⁺15]. The effect of wind wind speed on tremor amplitude becomes apparent for wind speeds above approximately 30 km/h. Note the different scales on the y-axes.

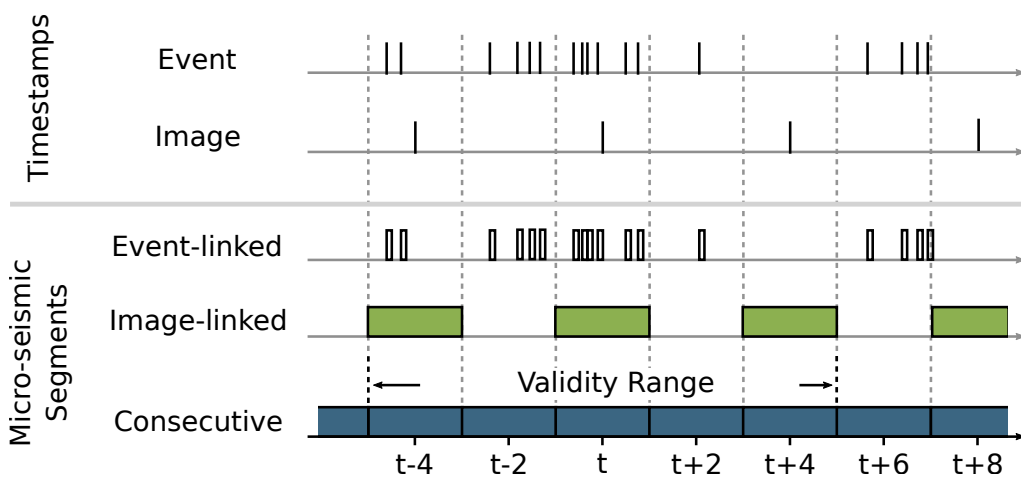


Figure 2.8 Illustration of micro-seismic segmentation. Event-linked segments are 10 second segments starting on event timestamps. Image-linked segments are two minute segments centered around an image timestamp. Consecutive segments are 2 minute segments sequentially extracted from the continuous micro-seismic signal.

2.4.2 Annotation

The continuous micro-seismic signals are segmented for annotation and evaluation. Figure 2.8 provides an overview of the three segmentation types, event-linked segments, image-linked segments and consecutive segments. Image-linked segments are extracted due the fact that a meaningful relation between seismic information and photos is only given in close temporal proximity. Therefore images and micro-seismic data are linked in the following way: For each image a 2 minute micro-seismic segment is extracted from the continuous micro-seismic signal. The micro-seismic segment's start time is set to 1 minute before the image timestamp. Event-linked segments are extracted based on the STA/LTA triggers from [WFM⁺18]. For each trigger 10 seconds following the timestamp of the trigger are extracted from the micro-seismic signal. Consecutive segments are 2 minute segments sequentially extracted from the continuous micro-seismic signal.

Only the image-linked segments are used during annotation, their label can however be transferred to other segmentation types by assigning the image-linked label to overlapping event-linked or consecutive segments. Image-linked and event-linked segments are used during data evaluation and classifier training. Consecutive segments are used for automatic classification on the complete dataset. Here, falsely classified segments are reduced by assigning each segment a validity range. A segment classified as mountaineer is only considered correct if the distance to the next (or previous) mountaineer is less than 5 minutes. This is based on an estimation of how long the mountaineers are typically in the audible range of the seismometer.

For mountaineer classification the required label is *mountaineer* but additional labels will be annotated which could be beneficial for classifier training and statistical analysis. These labels are *helicopters*, *rockfalls*, *wind*, *low visibility* (if the lens is partially obscured), and *lens flares*. The *wind* label applies to segments where the wind speed is higher than 30 km/h, which is the lower bound for noticeable wind impact as resulted from Sect. 2.4.1.

Figure 2.9 depicts the availability of image-linked segments per week during the relevant time frame. A fraction of the data is manually labeled by the authors, which is illustrated in Fig. 2.9. Two sets are created, a training set containing 5579 samples from the year 2016, and a test set containing 1260 data samples from 2017. The test set has been sampled randomly to avoid any human prejudice. For each day in 2017 four samples have been chosen randomly, which are then labeled and added to the test set. The training set has been specifically sampled to include enough training data for each category. This means for example that more mountaineers samples come from the summer period where the climbing route is most frequently used. The number of verified rockfalls and helicopters is non-representative and although helicopters can be manually identified from spectrograms the significance of these annotations is not given due the limited ground truth from the secondary

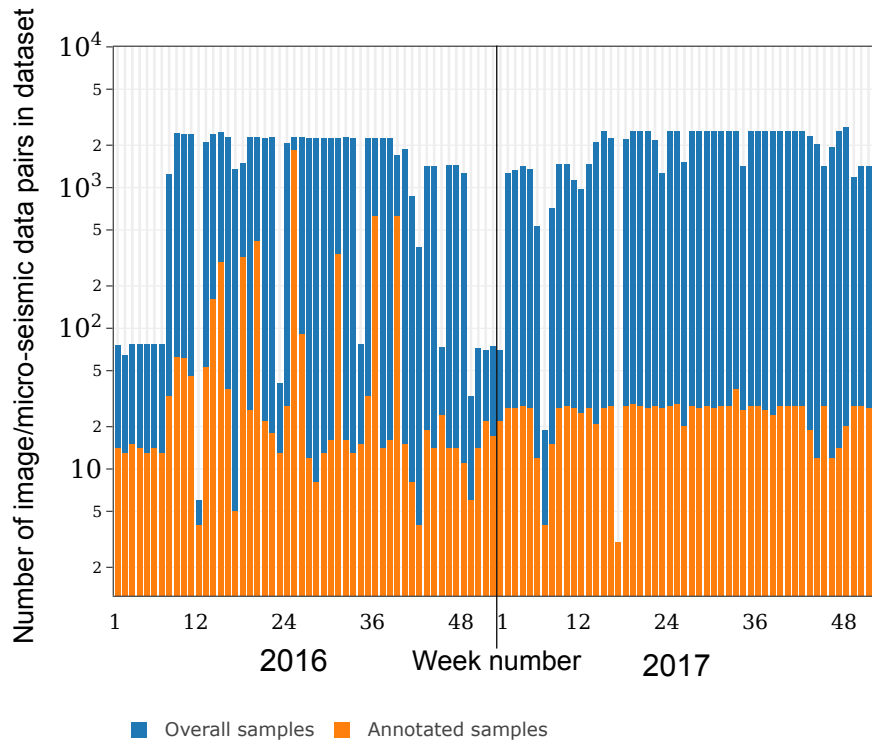


Figure 2.9 Number of image/micro-seismic data pairs in the dataset (dark blue) and in the annotated subset (light orange) displayed over the week number of the year 2016 and 2017. Note the logarithmic scale on the y-axis

source. Therefore, for the rest of this study we will focus on mountaineers for qualitative evaluation. For statistical evaluation we will however use the manually annotated helicopter and rockfall samples to exclude them from the analysis. The labels for all categories slightly differ for micro-seismic data and images since the type of sources which can be registered by each sensor differ. This means for instance that not every classifier uses all labels for training (for example a micro-seismic classifier cannot detect a lens flare). It also means that for the same time instance one label might apply to the image but not to the image-linked micro-seismic segment (for example mountaineers are audible but the image is partially obscured and the mountaineer is not visible). This becomes relevant in Sect. 2.5.3.4 when multiple classifiers are used for ensemble classification.

2.4.3 Data Types Selection

After the influences have been characterized the data type need to be selected which best describe each influence. The wind sensor delivers a continuous data stream and a direct measure of the external influence. In contrast, mountaineers, helicopters and rockfalls cannot directly be identified. A data type including information about these external influences needs to be selected.

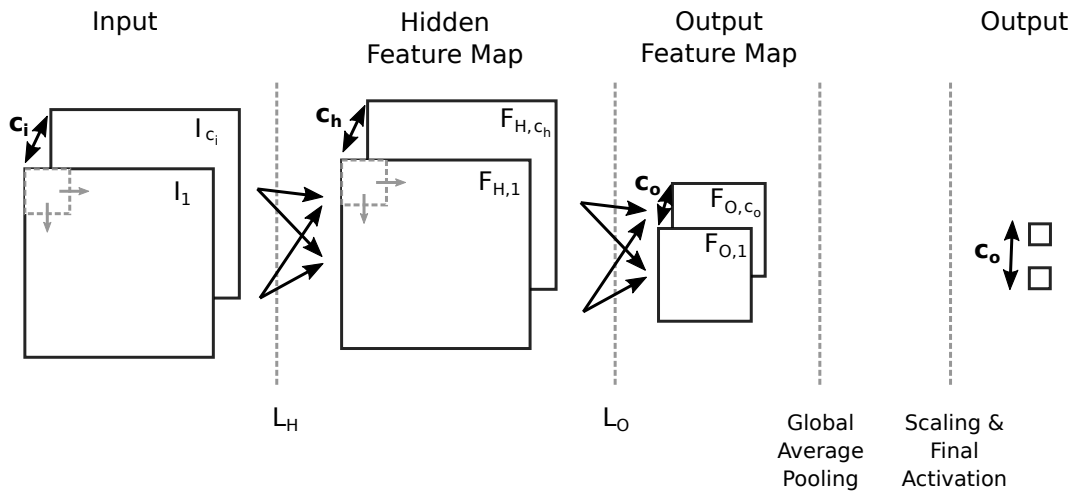


Figure 2.10 Simplified illustration of a convolutional neural network. An input signal, for example an image or spectrogram, with a given number of channels c_i is processed by a convolutional layer L_H . The output of the layer is a feature map with c_h channels. Layer L_O takes the hidden feature map as input and performs a strided convolution which results in the output feature map with reduced dimensions and number of channels c_o . Global average pooling is performed per channel and additional scaling and a final activation are applied.

Local observations, accommodation occupancy and flight data can be discarded for the use as classifier input since the data cannot be continuously collected. According to Sect. 2.4.1 it seems possible to identify mountaineers, helicopters and rockfalls from micro-seismic data. Moreover, mountaineers can also be identified from images. As a consequence, the data types selected to perform classification are micro-seismic data, images and wind data. The micro-seismic data used are the signals from the three components of the seismometer.

2.5 Classifier Selection and Training

The following section describes the classifier pre-selection, training, testing and how the classifiers are used to annotate the whole data stream as illustrated in Fig. 2.5 (b). First, a brief introduction to convolutional neural networks is given. If the reader is unfamiliar with neural networks we recommend to read additional literature [GBC16].

2.5.1 Convolutional Neural Networks

Convolutional neural networks have gained a lot of interest due to their advanced feature extraction and classification capabilities. A convolutional neural network contains multiple adoptable parameters which can be updated in an iterative optimization procedure. This fact makes them generically applicable to a large

range of datasets and a large range of different tasks. The convolutional neural network consists of multiple so-called convolutional layers. A convolutional layer transforms its input signal with c_i channels into c_h feature maps as illustrated in Fig. 2.10. A hidden feature map $F_{H,k}$ is calculated according to

$$F_{H,k} = g \left(\sum_{j=1}^{c_i} I_j * w_{k,j} + b_k \right)$$

where $*$ denotes the convolution operator, $g(\cdot)$ is a nonlinear function, I_j is an input channel, b_k is the bias related to the feature map $F_{H,k}$ and $w_{k,j}$ is the kernel related to the input image I_j and feature map $F_{H,k}$. Kernel and bias are trainable parameters of each layer. This principle can be applied to subsequent convolutional layers. Additionally, a strided convolution can be used which effectively reduces the dimension of a feature map as illustrated by L_1 in Fig. 2.10. In an all convolutional neural network [SDBR14] the feature maps of the output convolutional layer are averaged per channel. In our case, the number of output channels is chosen to be the number of event sources to be detected. Subsequent scaling and a final (non-linear) activation function are applied. If trained correctly each output represents the probability that the input contains the respective event source. In our case, this training is performed by calculating the binary cross-entropy between the network output and the ground truth. The error is backpropagated through the neural network and the parameters are updated. The training procedure is performed for all samples in the dataset and is repeated multiple epochs.

2.5.2 Classifier Selection

Multiple classifiers are available for the previously selected data types: wind data, images and micro-seismic data.

For wind data a simple threshold classifier can be used, which indicates wind influences based on the wind speed. For simplicity the classifier labels time periods with wind speed above 30 km/h as *wind*. For images a convolutional neural network is selected to classify the presence of mountaineers in the image. The image classifier architecture is selected from the large pool of available image classifiers [RDS⁺15]. For micro-seismic data, three different classifiers will be pre-selected: (i) a footstep detector based on manually selected features (standard deviation, kurtosis and frequency band energies) using a linear support vector machine (LSVM) similar to the detector used in [AMK18], (ii) a seismic event classifier adopted from [PGD18] and (iii) a non-geophysical event classifier which we call MicroseismicCNN. We reimplemented the first two algorithms based on the information from the respective papers. The third is a major contribution in this chapter and has been specifically designed to identify non-geophysical sources in micro-seismic data.

The proposed convolutional neural network (CNN) to identify non-geophysical

Layer	stride	output channels
Conv2D + BatchNorm + Linear	1	32
Conv2D + BatchNorm + ReLU	2	32
Dropout	-	32
Conv2D + BatchNorm + ReLU	2	32
Dropout	-	32
Conv2D + BatchNorm + ReLU	1	32
Conv2D + ReLU	1	32
Dropout	-	32
Conv2D + ReLU	1	1
Global Average Pooling	1	1
Dropout	-	1
Conv2D	1	1
Sigmoid Activation	-	1

Table 2.1 Layout of the proposed non-geophysical event classifier, consisting of multiple layers which are executed in sequential order. The convolutional neural network consists of multiple 2D convolutional layers (Conv2D) with batch normalization (BatchNorm) and rectified linear units (ReLU). Dropout layers are used to minimize overfitting. The sequence of global average pooling layer, a scaling layer and the sigmoid activation compute one value between 0 and 1 resembling the probability of a detected mountaineer.

sources in micro-seismic signals uses a time-frequency signal representation as input and consists of 2D convolutional layers. Each component of the time-domain signal, sampled at 1 kHz, is first offset-compensated and then transformed with a Short-Time Fourier Transformation (STFT). Subsequently, the STFT output is further processed by selecting the frequency range from 2 to 250 Hz and subdividing it into 64 linearly-spaced bands. This time-frequency representation of the three seismometer components can be interpreted as 2D signal with three channels, which is the networks input. The network consists of multiple convolutional, batch normalization and dropout layers, as depicted in Table 3.3. Except for the first convolutional layer, all convolutional layers are followed by batch normalization and Rectified Linear Units (ReLU) activation. Finally, a set of global average pooling layer, dropout, trainable scaling (in form of a convolutional layer with kernel size 1) and sigmoid activation reduces the features to one value representing the probability that a mountaineer is in the micro-seismic signal. In total the network has 30,243 parameters. In this architecture multiple measures have been taken to minimize overfitting: the network is all-convolutional [SDBR14], batch normalization [IS15] and dropout [SHK⁺14] are used and the size of the network is small compared to recent audio classification networks [HCE⁺16].

2.5.3 Training and Testing

We will evaluate the micro-seismic algorithms in two scenarios in Sect. 2.7.1. In this section, we describe the training and test setup for the two scenarios as well as for image and ensemble classification. In the first scenario event-linked segments are classified. In the second scenario the classifiers on image-linked segments are compared. The second scenario stems from the fact that the characterization from Sect. 2.4.1 suggested that using a longer temporal input window could lead to a better classification because it can capture more characteristics of a mountaineer. Training is performed with the annotated subset from Sect. 2.4 and a random 10 % of the training set are used as validation set, which is never used during training. For the non-geophysical and seismic event classifiers the number of epochs has been fixed to 100 and for the image classifier to 20. After each epoch the F1 score of the validation set is calculated and based on it the best performing network version is selected. The F1 score is defined as

$$\text{F1 score} = \frac{2 \cdot \text{true positive}}{2 \cdot \text{true positive} + \text{false negative} + \text{false positive}}$$

The threshold for the network's output is determined by running a parameter search with the validation set's F1 score as metric. Training was performed in batches of 32 samples with the ADAM [KB14] optimizer and cross-entropy loss. The Keras [Cho15] framework with a TensorFlow backend [AAB⁺15] was used to implement and train the network. The authors of the seismic event classifier [PGD18] provide TensorFlow source code, but to keep the training procedure the same it was re-implemented with the Keras framework. The footstep detector is trained with scikit-learn [PVG⁺11]. Testing is performed on the test set which is independent of the training set and has not been used during training. The metrics error rate and F1 score are calculated.

It is common to do multiple iterations of training and testing to get the best performing classifier instance. We perform a preliminary parameter search to estimate the number of iterations. The estimation takes into account the number of training types (10 different classifiers need to be trained) given the limited processing capabilities. As a result of the search, we train and test 10 iterations and select the best classifier instance of each classifier type to evaluate and compare their performances in Sect. 2.7.

The input of the micro-seismic classifiers must be variable to be able to perform classification on event-linked segments and image-linked segments. Due to the principle of convolutional layers, the CNN architectures are independent of the input size and therefore no architectural changes have to be performed. The footstep detector's input features are averaged over time by design and are thus also time-invariant.

2.5.3.1 Event-linked Segments Experiment

Literature suggests that STA/LTA cannot distinguish geophysical sources from non-geophysical sources [All78]. Therefore the first micro-seismic experiment investigates if the presented algorithms can distinguish events induced by mountaineers from other events in the signal. The event-linked segments are used for training and evaluation. The results will be discussed in Sect. 2.7.1.

2.5.3.2 Image-linked Segments Experiment

In the second micro-seismic experiment the image-linked segments will be used. Each classifier is trained and evaluated on the image-linked segments. The training parameters for training the classifiers on image-linked segments are as before but additionally data augmentation is used to minimize overfitting. Data augmentation includes random circular shift and random cropping on the time axis. Moreover, to account for the uneven distribution in the dataset, it is made sure that during training the convolutional neural networks see one example of a mountaineer every batch. The learning rate is set to 0.0001, which was determined with a preliminary parameter search. The classifiers are then evaluated on the image-linked segments.

To be able to compare the results of the classifiers trained on image-linked segments to the classifiers trained on event-linked segments (Sect. 2.5.3.1), the classifiers from Sect. 2.5.3.1 will be evaluated on the image-linked segments as well. The metrics can be calculated with the following assumption: If any of the event-linked segments which are overlapping with an image-linked segment are classified as mountaineer, the image-linked segment is considered to be classified as mountaineer as well.

The results will be discussed in Sect. 2.7.1.

2.5.3.3 Image Classification

Since convolutional neural networks are a predominant technique for image classification, a variety of network architectures have been developed. For this study, the MobileNet [HZC⁺17] architecture is used. The number of labeled images is small in comparison to the network size (approx. 3.2M parameters) and training the network on the Matterhorn images will lead to overfitting on the small dataset. To reduce overfitting a MobileNet implementation which has been pre-trained on ImageNet [DDS⁺09], a large-scale image dataset, will be used. Retraining is required since ImageNet has a different application focus than this study. The climbing route, containing the subject of interest, only covers a tiny fraction of the image and rescaling the image to 224x224, the input size of the MobileNet, would lead to vanishing mountaineers (compare Fig. 2.4). However, the image size cannot be chosen arbitrarily large since a larger input requires more memory and results in a larger runtime. To overcome

this problem the image has been scaled to 448x672 pixels and although the input size differs from the pretrained version network retraining still benefits from pre-trained weights. Data augmentation is used to minimize overfitting. For data augmentation each image is slightly zoomed in and shifted in width and height. The network has been trained to detect 5 different categories. In this chapter only the metrics for mountaineers are of interest for the evaluation and the metrics for the other labels are discarded in the following. However, all categories are relevant for a successful training of the mountaineer classifier. These categories consist of mountaineer, low visibility (if the lens is partially obscured), lens flare, snowy (if the seismometer is covered in snow) and bad weather (as far as it can be deduced from the image).

2.5.3.4 Ensemble

In certain cases, a sensor cannot identify a mountaineer although there is one, for example the seismometers cannot detect when the mountaineer is not moving or the camera does not capture the mountaineer if the visibility is low. The usage of multiple classifiers can be beneficial in these cases. In our case micro-seismic and image classifier will be jointly used for mountaineer prediction. Since micro-seismic labels and image labels are slightly different, as discussed in Sect. 2.4.2, the ground truth labels must be combined. For a given category, a sample is labeled true if any of micro-seismic or image labels are true (logical disjunction). After individual prediction by each classifier the outputs of the classifiers are combined similarly and can be compared to the ground truth.

2.5.4 Optimization

Due to potential human errors during data labeling, the training set has to be regarded as a weakly-labeled dataset. Such datasets can lead to a worse classifier performance. To overcome this issue a human-in-the-loop approach is followed where a preliminary set of classifier is trained on the training set. In the next step, each sample of the dataset is automatically classified. This procedure produces a number of true positives, false positives and true negatives. These samples are then manually relabeled and the labels for the dataset are updated based on human review. The procedure is repeated multiple times. This does however not completely avoid the possibility of falsely labeled samples in the dataset, since the algorithm might not find all human-labeled false negatives, but it increases the accuracy significantly. The impact of false labels on classifier performance will be evaluated in Sect 2.7.1.

	Error Rate	F1 Score
Event-linked Segments		
Footstep Detector (Events)	0.1702	0.7692
Seismic Event Classifier (Events)	0.1250	0.8291
MicroseismicCNN (Events)	0.0641	0.9062
Image-linked Segments		
Footstep Detector (Events)	0.0706	0.5389
Seismic Event Classifier (Events)	0.0540	0.6047
MicroseismicCNN (Events)	0.0309	0.731
Footstep Detector	0.0952	0.52
Seismic Event Classifier	0.0313	0.7383
MicroseismicCNN	0.0096	0.9167
Image Classifier	0.0088	0.9134
Ensemble	0.0079	0.9383

Table 2.2 Results of the different classifiers. The addition "(Events)" labels the classifier versions trained on event-linked segments

2.6 Automatic Classification

In Sect. 2.7.1 it will be shown that the best set of classifiers are the ensemble of image classifier and MicroseismicCNN. Therefore, the trained image classifiers and MicroseismicCNN classifier are used to annotate the whole time period of collected data to quantitatively assess the impact of mountaineers. The image classifier and the MicroseismicCNN will be used to classify all the images and micro-seismic data, respectively. The consecutive segments and images are used for prediction. To avoid false positives we assume that a mountaineer requires a certain amount of time to pass by the seismometer as illustrated in Fig. 2.8, therefore a mountaineer annotation is only considered valid if its minimum distance to the next (or previous) mountaineer annotation is less than 5 minutes. Subsequently, events within time periods classified as mountaineer are removed and the event count per hour is calculated.

2.7 Evaluation

In the following the results of the different classifiers experiments described in Sect. 2.5.3 will be presented to determine the best set of classifiers. Furthermore, in Sect. 2.7.2 and Sect. 2.7.3 results of the automatic annotation process (Sect. 2.6) will be used to evaluate the impact of external influences on the whole dataset.

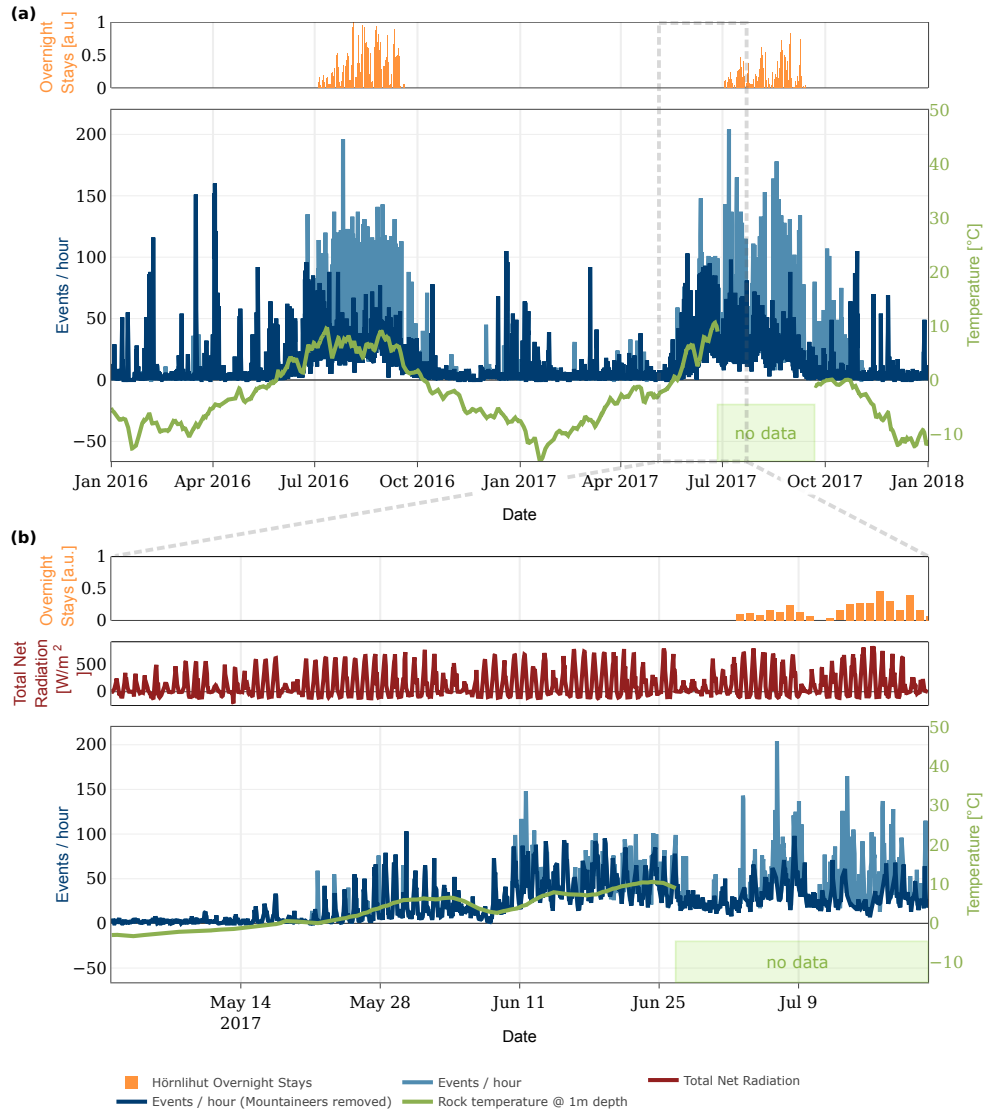


Figure 2.11 Event count, hut occupancy and rock temperature over time. (a) For the years 2016/2017 and (b) for a selected period during defreezing of the rock. The event rate from [WFM⁺18] is illustrated in light blue and the rate after removal of mountaineer induced events in dark blue. The strong variations in event rate correlate with the presence of mountaineer, hut occupancy and in (b) with the total net radiation. The impact of mountaineers is significant after July 9th and event detection analysis becomes unreliable.

False Labels (%)	25	12.5	6.25	3.125	0
F1 score (mean)	0.7953	0.8633	0.8761	0.8835	0.8911
Error rate (mean)	0.0208	0.0149	0.0139	0.0129	0.0122

Table 2.3 Influence of falsely labeled data points on the test performance. Shown are the mean values over 10 training/test iterations.

2.7.1 Classifier Evaluation

The results of the classifier experiments (Table 2.2) show that the footstep detector is the worst at classifying mountaineers with an error rate of 0.1702 on event-linked segments and 0.0952 on image-linked segments. Both convolutional neural networks score a lower error rate on image-linked segments of 0.0096 (MicroseismicCNN) and 0.0313 (Seismic Event Classifier). For the given dataset our proposed MicroseismicCNN network outperforms the seismic event classifier, in both the event-linked segment experiment as well as the image-linked segment experiment. The MicroseismicCNN using a longer input window (trained on image-linked segments) is comparable to classification on images and outperforms the classifier trained on event-linked segments. When combining image and micro-seismic classifiers the best results can be achieved.

The number of training/test iterations that were run for each classifier has been set to 10 through a preliminary parameter estimation. To validate our choice we have evaluated the influences of the number of experiments for only one classifier. The performance of the classifier is expected to depend on the number of training/test iterations (more iterations means a better chance of selecting the best classifier). However, the computing time is increasing linearly with increasing number of iterations. Hence, a reasonable trade off between the performance of the classifier and the computing time is desired to identify the ideal number of iterations. Figure 2.12 represents the statistical distribution of the classifier's performance for different number of training/test iterations. Each boxplot is based on ten independent sets of training/test iterations. While the box indicates the interquartile range (IQR) with the median value in orange, the whisker on the appropriate side is taken to $1.5 \times IQR$ from the quartile instead of the maximum or minimum if either type of outlier is present. Beyond the whiskers, data are considered outliers and are plotted as individual points. As can be seen in Fig. 2.12, the F1 score saturates at 9 iterations. Therefore our choice of 10 iterations is a reasonable choice.

In Sect. 2.5.4 the possibility of falsely labeled training samples has been discussed. As expected, our evaluation in Table 2.3 indicates that falsely labeled samples have an influence on the classification performance since the mean performances are worse for a high percentage of false labels.

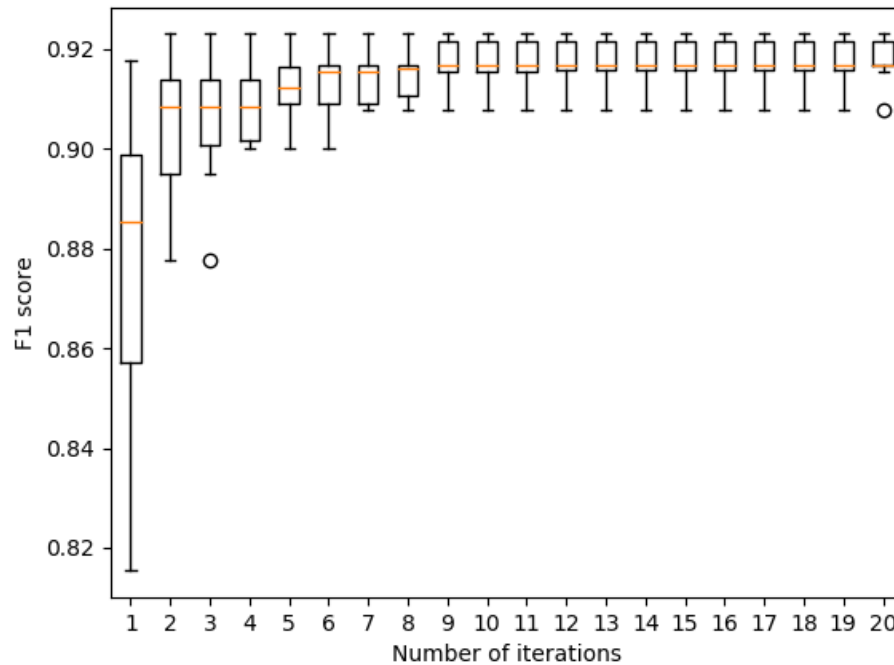


Figure 2.12 The statistical distribution of the classifier’s performance for different number of training/test iterations is illustrated. Each boxplot is based on ten independent sets of training/test iterations. The F1 score saturates after 9 iterations and validates our choice of 10 iterations.

2.7.2 Statistical Evaluation

The annotated test set from Sect. 2.4.2 and the automatically annotated set from Sect. 2.6 are used for a statistical evaluation involving the impact of external influences on micro-seismic events. Only data from 2017 is chosen since wind data is not available for the whole training set due to a malfunction of the weather station in 2016. The experiment from [WFM⁺18] provides STA/LTA event triggers 2017. Table 2.4 shows statistics for several categories, which are 3 external influences and one category where none of the 3 external influences are annotated (declared as "Unknown"). For each category, the total duration of all annotated segments is given and how many events per hour are triggered. It becomes apparent that mountaineers have the biggest impact on the event analysis. Up to 105.9 events per hour are detected on average during time periods with mountaineer activity, while during all other time periods the average ranges from 9.09 to 13.12 events per hour. This finding supports our choice to mainly focus on mountaineers in this chapter and shows that mountaineers have a strong impact on the analysis. As a consequence, a high activity detected by the event trigger does not correspond to a high seismic activity, thus relying only on this kind of event detection may lead to a false interpretation. From the automatic section in Table 2.4 it can be deduced that

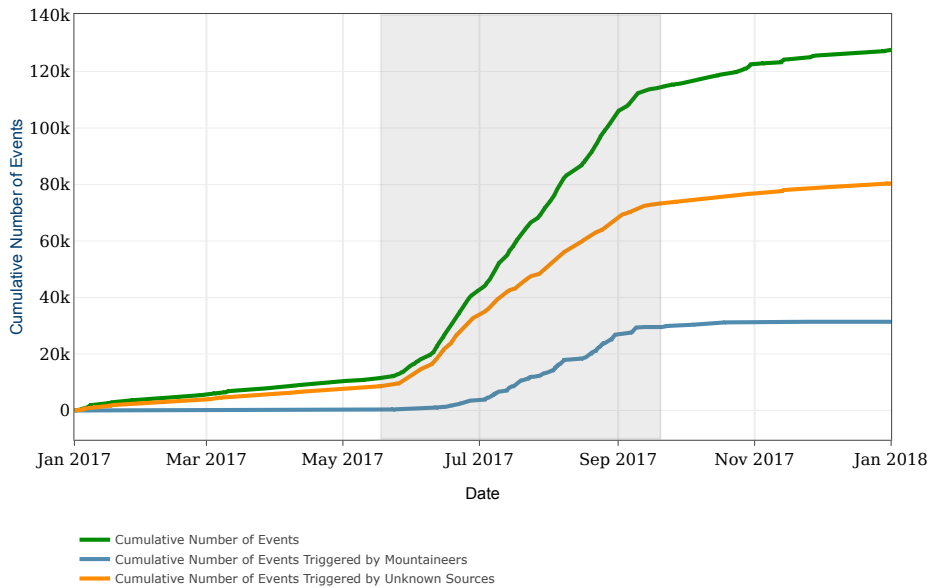


Figure 2.13 Illustration of the cumulative number of events triggered by the STA/LTA event detector for all events, for events triggered by mountaineers and for events triggered by unknown sources. The results presented in this chapter were used to annotate the events. The time period during which the rock temperature in 1 m depth is above 0°C is shaded in gray.

the average number of triggered events per hour for times when the signal is influenced by mountaineers is an approximately 9x increase in comparison to periods without annotated external influences. The effect of wind influences on event rate is not as clear as the influence of mountaineers. The values in Table 2.4 indicate a decrease of events per hour during wind periods, which will be briefly discussed in Sect. 2.8.2.

As can be seen in Fig. 2.13, events are triggered over the course of the whole year whereas events that are annotated as coming from mountaineers occur mainly during the summer period. The main increase in event count occurs during the period when the rock is unfrozen which unfortunately coincides with the period of mountaineer activity. Therefore it is important to account for the mountaineers. However, even if the mountaineers are not considered the event count increases significantly during the unfrozen period. The interpretation of these results will not be part of this study but they are an interesting topic for further research.

2.7.3 Automatic Annotation in a Real-World Scenario

The results of applying the ensemble classifier to the whole dataset is visualized for two time periods in Fig. 2.11. The figure depicts the event count per hour before and after removing periods of mountaineer activity, as well as the rock temperature, the overnight stays at the Hörnlihut and the total net

		Unknown	Mountaineer	Wind	Helicopter
Manual	duration (hours)	28.87	1.9	6.6	3.73
	mean number of events per hour	10.6	95.26	11.21	13.12
Automatic	duration (hours)	6832.3	296.53	1364.2	-
	mean number of events per hour	11.76	105.9	9.09	-

Table 2.4 Statistics of the manually and automatically annotated set of 2017 per annotation category. "Unknown" represents the category when none of the other categories could have been identified. Given are the total duration of annotated segments per category and the mean number of STA/LTA events per category.

radiation. From Fig. 2.11 (a) it becomes apparent that the mountaineer activity is mainly present during summer and autumn. An increase is also visible during increasing hut overnight stays. During winter and spring only few mountaineers are detected but some activity peaks remain. By manually review we were able to discard mountaineers as cause for most of these peaks, however further investigation is needed to explain their occurrence.

Figure 2.11 (b) focuses on the defreezing period. The zero-crossing of the rock temperature has a significant impact on the event count variability. A daily pattern becomes visible starting around the zero-crossing. Since few mountaineers are detected in May these can be discarded as the main influence for these patterns. The total net radiation however indicates an influences of solar radiation on the event count. Further in-depth analysis is needed but this examples shows the benefits of a domain-specific analysis, since the additional information gives an intuition of relevant processes and their interdependencies. After July 9th, the impact of mountaineers is significant and the event detection analysis becomes unreliable. Different evaluation methods are required to mitigate the influence of mountaineers during these periods.

Figure 2.14 depicts that mountaineer predictions and hut occupancy correlate well, which indicates that the classifiers work well. The discrepancy in the first period of each summer needs further investigation. With the annotations for the whole timespan it can be estimated that from all events detected in [WFM⁺18], approximately 25% originate in time periods with mountaineer activity and should therefore not be regarded as originating from geophysical sources.

2.8 Discussion

2.8.1 Classification of Negative Examples

The previous section has shown that a certain degree of understanding of the scenario and data collected is nevertheless necessary in order to achieve a significant analysis. The effort in creating an annotated data subset, despite being time and labor consuming, is an overhead but as we show can be outweighed by the benefits of better analysis results. For data annotation two distinct approaches can be followed: Annotating the phenomena of interest (positive examples) or annotating the external influences (negative examples). Positive examples, used in [YLW⁺18, RMB⁺14, KG17], inherently contain a limitation as this approach requires that events as well as influencing factors must be identified and identifiable in the signal of concern. This is especially hard where no ground truth information except (limited) experience by professionals is available. Therefore, the strategy presented in this chapter to create an annotated dataset using negative examples is advisable to be used. It offers to perform cross checks if certain patterns can be found in different sensor/data types and in many cases the annotation process can be performed

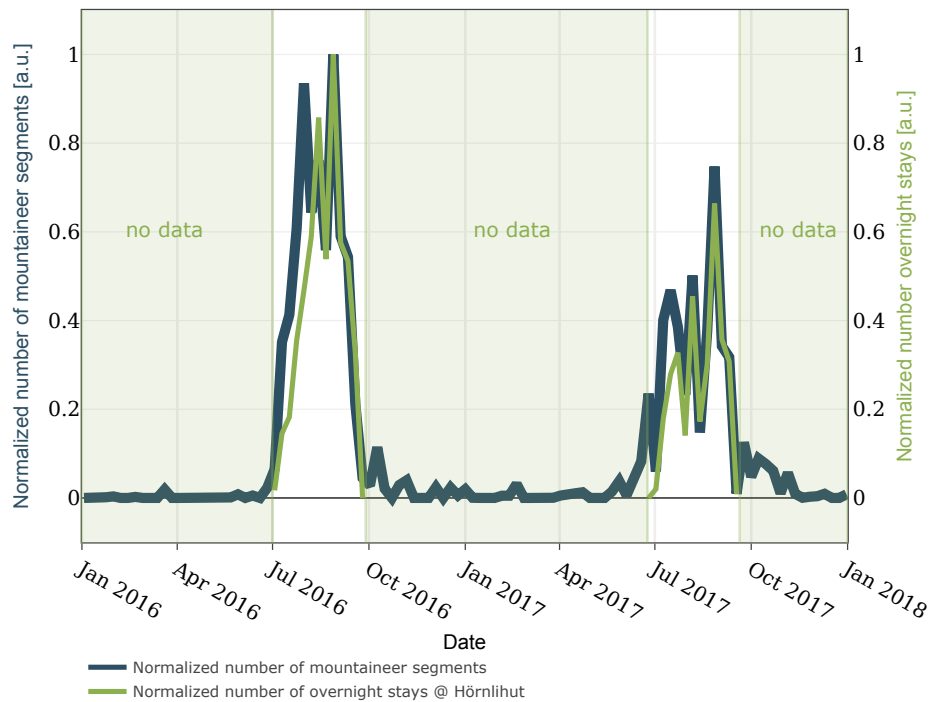


Figure 2.14 Correlation of mountaineer activity and hut occupancy. The normalized number of mountaineer segments per week and the normalized number of overnight stays at the Hörnlihut per week plotted over time.

by non-experts. Also, additional sensors allow to directly quantify possible influence factors. The detour required by first classifying negative examples and then analyzing the phenomena of interest offers further benefits: In cases where the characteristics of the phenomena of interest are not known in advance (no ground truth available) and in cases where a novel analysis method is to be applied or when treating very-long-term monitoring datasets working only on the primary signal of concern is hard and error margins are likely to be large. In these cases it is important to take into account all knowledge available including possible negative examples and it is significant to automate as much as possible using automatic classification methods.

2.8.2 Multi-Sensor Classification

In Sect. 2.5 multiple classifiers for different sensors have been presented. The advantages of classifying micro-seismic signals are that continuous detection is possible and that no additional sensors are required. The classification accuracy of the convolutional neural network and the image classifier presented are comparable. Classification of time-lapse images however has the disadvantage of a low time resolution proportional to the capture frequency, for example a maximum of 15 images per hour in our example. Continuous video recording

could close this gap at the cost of requiring a more complex image classifier, the size of the data and more higher processing times, which are likely infeasible. The main advantage of images is that they can be used as additional independent sensor to augment and verify micro-seismic recordings. First, images can be used for annotations and second they can be used in an ensemble classifier to increase the overall accuracy. The different modalities strengthen the overall meaningfulness and make the classifier more robust. Table 2.4 shows that during windy segments less events are triggered than in periods that cannot be categorized ("unknown" category). A possible explanation is that the micro-seismic activity is superimposed by broadband noise originating in the wind. For these time segments a variable trigger sensitivity [WDF08] or a different event detection algorithm can improve the analysis. Better shielding the seismometer from the wind would reduce these influences significantly but the typical approach in seismology to embed it into the ground under a substantial soil column is next to impossible to implement in steep bedrock and perennially frozen ground as found on our case-study field site. Nonetheless, Table 2.4 gives an intuition that our method performs well since the statistical distribution of manually and automatically annotated influences sources is similar. We therefore conclude that with our method presented it is possible to quantify the impact of external influences on a long-term scale and across variable conditions.

2.8.3 Feature Extraction

In Sect. 2.4.1 the different characteristics of event sources have been discussed. The characteristic features can be used to identify and classify each source type. The convolutional neural network accomplishes the task of feature extraction and classification simultaneously by training on an extensive annotated dataset. An approach without the requirement of an annotated dataset would be to manually identify the characteristics and then design a suitable algorithm to extract the features. For example the helicopter pattern in 2.6 (c) shows distinct energy bands indicating the presence of a fundamental frequency plus harmonics. These features could be traced to identify, model and possibly localize a helicopter [ELBA17] with the advantage of a relaxed dataset requirement. The disadvantage would be the requirement of further expertise in the broad field of digital signal processing and modeling as well as more detailed knowledge on each such phenomena class of interest. Also, it is likely that such an approach would require extensive sensitivity analysis to be performed alongside with modeling. Moreover, if the algorithm is handcrafted by using few examples it is prone to overfitting based on these examples (see also the next subsection). This problem of overfitting exist as well for algorithm training and can be solved by using more examples, however, it is easier to annotate a given pattern (with the help of additional information) than understanding its characteristics and thus the time- and labor-consuming task of annotation can be outsourced in the case of machine learning. Fig. 2.6 indicates that little anthropogenic noise (a) has less broadband background noise than wind (d) and the impulses occur in a

different frequency band. However, the signal plots show a similar pattern. To identify wind from micro-seismic data manually one could utilize a frequency-selective event detector although it is not clear if this pattern and frequency range is representative for every occurrence of wind and if all non-wind events could be excluded with such a detector. Using a dedicated wind sensor for identification of wind periods as presented in this study overcomes these issues with the drawback of an additional sensor which needs to be installed and maintained and that during failure of the additional sensor no annotation can be performed.

2.8.4 Overfitting

A big problem with machine learning methods is overfitting due to too few data examples. Instead of learning representative characteristics the algorithm memorizes the examples. In our work overfitting is an apparent issue since the reference dataset is small as described in Sect. 2.4.2. As explained in the previous sections multiple measures have been introduced to reduce overfitting (data augmentation, few parameters, all convolutional neural network, dropout). The test set has been specifically selected to be from a different year to exclude that severe overfitting affects the classifier performance. The test set includes examples from all seasons, day and night time and is thus assumed representative for upcoming, never-seen-before data. However, overfitting might still exist in the sense that the classifier is optimized for one specific seismometer. Generalization to multiple seismometers still needs to be proven since we did not test the same classifier for multiple seismometers, which might differ in their specific location, type or frequency response. This will be an important study for the future since it will reduce the dataset collection and training time significantly if a new seismometer is deployed.

2.8.5 Outlook

This chapter has only focused on identifying external influences, what we have shown to be a prerequisite for micro-seismic analysis. Future work lies in finding and applying specific analytic methods, especially finding good parameter sets and algorithms for each context. Additionally, the classifier could be extended to include helicopters as well as geophysical sources such as rockfalls. A disadvantage of the present method is the requirement of a labeled dataset. Semi-supervised or unsupervised methods [KYDH11] as well as one- or few-shot classification methods [FFP06] could provide an alternative to the presented training concept without the requirement of a large annotated dataset. We will present some of these methods in the upcoming chapters, especially in Chapter 4.

2.9 Conclusion

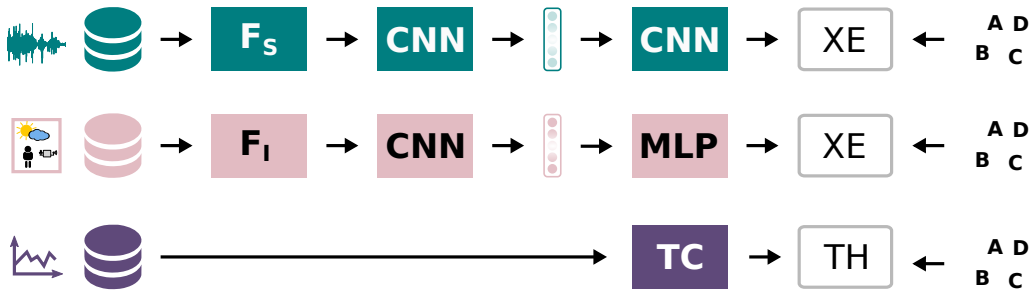


Figure 2.15 Data types, models and optimizations used in this chapter. Machine learning-based information extraction from images, seismic data and time-series data using data specific frontend (F_S, F_I), convolutional neural networks (CNN), multi-layer perceptron (MLP) and threshold classifiers (TC). The models are independently optimized with cross-entropy loss (XE) and threshold selection (TH) using a set of labels.

In this chapter we have presented a strategy to evaluate the impact of external influences on a micro-seismic measurement by categorizing the measurements with the help of additional sensors and information. With this knowledge a method to classify mountaineers has been presented. Figure 2.15 conceptually summarizes the elements of the information extraction system used in this chapter. We have shown how additional sensors can be beneficial to isolate the information of interest from unwanted external influences and provide a ground truth in a long-term monitoring setup. Moreover, we have presented a mountaineer detector, implemented with a convolutional neural network, which scores an error rate of only 0.96 % (F1 score: 0.9167) on micro-seismic signals and a mountaineer detector ensemble which scores an error rate of 0.79 % (F1 score: 0.9383) on images and micro-seismic data. The classifiers outperform comparable algorithms. Their application to a real-world, multi-sensor, multi-year micro-seismic monitoring experiment showed that time periods with mountaineer activity have a approximately 9x higher event rate and that approximately 25% of all detected events are due to mountaineer interference. Finally, the findings of this chapter show that an extensive, systematic identification of external influences is required for a quantitative and qualitative analysis on long-term monitoring experiments. In the next chapter, we will focus on how this knowledge can be integrated into the monitoring system by using the mountaineer classifier on the seismic sensor device in the context of natural hazard warning.

2.A Appendix





Related Publications

Systematic identification of external influences in multi-year microseismic recordings using convolutional neural networks

Matthias Meyer, Samuel Weber, Jan Beutel, and Lothar Thiele

Earth Surface Dynamics, vol. 7, no. 1, pp. 171–190, Feb. 2019

Contributions: Matthias Meyer developed the concept and discussed it with Samuel Weber, Jan Beutel and Lothar Thiele. Matthias Meyer and Samuel Weber developed the code and maintained field site and data together with Jan Beutel. Matthias Meyer prepared and performed the experiments and evaluated the results with Samuel Weber. Matthias Meyer prepared the manuscript as well as the visualizations with contributions from all co-authors.

	Paper	10.5194/esurf-7-171-2019
	Code	10.5281/zenodo.1321176
	Dataset	10.5281/zenodo.1320835
	Project Webpage	matthiasmeyer.xyz/noise-on-matterhorn/

3

Hazard monitoring using machine learning and edge devices

In the previous chapter, we discussed how to use a combination of machine learning, auxiliary sensors and human in the loop to identify unwanted external influences in micro-seismic recordings. In this chapter, we focus on moving the knowledge generation process to the edge, which means computing a pretrained model for seismic event detection on a custom-designed, event-triggered geophone sensor. We will proceed in two steps. First, in Section 3.1, we focus on architectural optimizations of an existing convolutional neural network and evaluate these optimizations on an acoustic event dataset. Then, in Section 3.2, we modify the acoustic event classifier to work with seismic data and train it using the dataset compiled in Chapter 2. We further optimize the execution of the model on a single-core microcontroller and evaluate its performance with field data generated by a deployment of an array of event-triggered geophones.

In natural hazard warning systems fast decision making is vital to avoid catastrophes. Advanced data processing and decision making at the edge of a wireless sensor network promises fast response times but is limited by the availability of energy, data transfer speed, processing and memory constraints. In the area of distributed seismic sensing, the combination of algorithms with a high classification rate and resource-constraint embedded systems is essential. However, this combination has not been demonstrated for seismic event classification. In recent years, convolutional neural networks have proven to have a high classification accuracy for audio event detection, which is closely related to seismic event detection. Unfortunately, these algorithms for acoustic event detection have a high memory and computational demand and are not suited for execution at the network edge. Additionally, the measurement hardware is not yet optimized for our application scenario in high-alpine environments. Off-the-shelf geophones and digitizers are usually not designed for long-term,

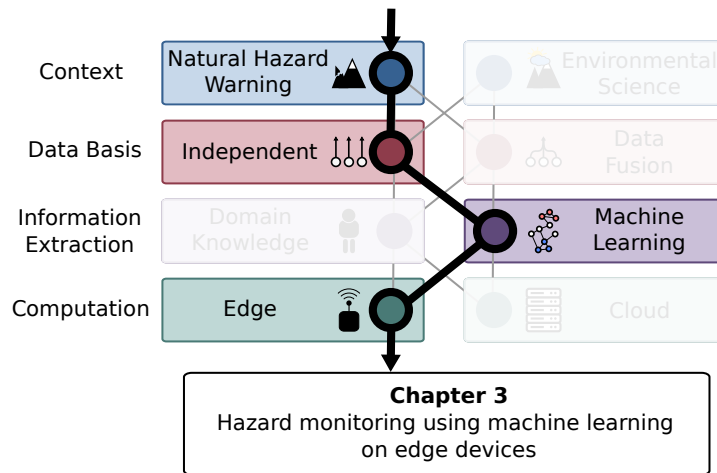


Figure 3.1 This chapter's position in the context of the dissertation

low-power applications in harsh environments, making them difficult to deploy in large quantities in our setup.

In this chapter, we present the path towards a realization of a wireless sensor network for hazard monitoring based on an array of event-triggered single-channel micro-seismic sensors with advanced signal processing and characterization capabilities based on a novel co-detection technique. In Section 3.2.4, we present an ultra-low power, threshold-triggering circuit paired with on-demand digital signal acquisition capable of extracting relevant information exactly and efficiently at times when it matters most and consequentially not wasting precious resources when nothing can be observed. In Section 3.2.5, hand we utilize machine-learning-based classification implemented on low-power, off-the-shelf microcontrollers to avoid false positive warnings and to actively identify humans in hazard zones. The sensors' response time and memory requirement is substantially improved by (i) applying architectural optimization to a convolutional neural network (Section 3.1), (ii) quantizing it (Section 3.2.5.4) and (iii) pipelining its inference (Section 3.2.6). In this way, convolutional neural networks that would not run unmodified on a memory constrained device can be executed in real-time and at scale on low-power embedded devices. A field study with our system was running on the rockfall scarp of the Matterhorn Hörnligrat at 3500 m a.s.l. from 08/2018 until 02/2020.

3.1 Architectural Optimizations For Event Classification

In this section we present architectural optimizations using a reference audio event classifier based and verify them on an audio event dataset. These findings serve as a basis for the seismic event classifier used in the upcoming section.

Many applications for sensor networks, cyber-physical system or Internet-of-Things require a low power consumption for long-term autonomous operation. Local preprocessing and classifying data at the edge nodes can be a solution to reduce data transmission and therefore, to reduce energy consumption. In addition, such an approach can avoid that enormous amounts of data need to be communicated to and processed by a centralized data analysis infrastructure.

The detection of acoustic events is a typical example: Instead of streaming audio through the network to perform server-side event detection, the acoustic events of interest can be pre-detected directly at the sensing device reducing the network's data throughput and the response time significantly. The accurate detection and classification of individual acoustic events from a sound-emitting environment is of interest for many application such as surveillance [CBM07] or environmental monitoring [GBG⁺12]. The low-power embedded devices used for such systems, however, come with very stringent memory and throughput limitations. These resource constraints impose severe limits to the complexity of suitable event detection algorithms.

Hardware platforms for battery-operated devices have stringent power requirements. On such low-power devices which are limited to a few 100 mW, like the ARM Cortex M7 series, the overall on-chip storage is typically limited to less than 2 MB and the available digital signal processing performance is limited to a few 100 millions multiply-accumulate operations per second even on the most advanced components [NXP16, STM16]. To achieve maximum energy efficiency while using CNNs, System-on-Chips (SoCs) with hardware accelerators for CNN workload or more generally 2D convolutions can be considered. Such platforms can provide speed-ups by a factor of around 100× and an improved energy efficiency of about 40× [CB15, CGM⁺15]. Such system perform optimal if the CNN comprises a simple and structured architecture. While this concept can provide a relief on the admissible computational effort, the strong limitations on available memory remain because any external memory would deteriorate the device's energy efficiency substantially.

For acoustic event detection different algorithms have been presented based on Non-Negative Matrix Factorization [KTKS16], Hidden Markov Models [ZZL⁺08] or Recurrent Neural Networks [PHV16]. Like in many other machine learning applications, CNNs have been proven to be the key for high classification accuracy. The advantage of such an architecture for acoustic event detection is its inherent inclusion of a temporal neighborhood since acoustic events are strongly characterized by temporal changes. CNNs have been used mainly for speech recognition [ArMJ⁺14] or music [DS14] classification tasks. These algorithms are all computationally expensive, and come at the expense of

having a huge number of parameters [TGPG16], resulting in computational and memory overhead. Implementing such a network on mobile devices or sensor nodes is difficult due to memory and computational restrictions on these devices. Recently, a convolutional neural network (CNN)-based approach has been proposed for acoustic event detection [TGPG16] using a network design adapted from image classification [SZ14]. It has been shown that this approach outperforms previous state-of-the-art approaches by a large margin. Unfortunately, the proposed algorithm also comes at the expense of having a huge memory requirement due to the huge number of parameters. For image classification a way to reduce the complexity of CNN-based classification systems has been presented [SDBR14]. Applying these complexity reduction is used in this chapter to build a highly-accurate CNN capable of running on embedded platforms with limited resources.

In this context, this section contains the following contributions:

- A convolutional neural network for acoustic event classification is selected as basis for a seismic event classifier. Its architecture is optimized towards high classification accuracy while reducing the memory requirements and number of operations.
- The benefits of optimization are experimentally verified by comparing it to the unmodified convolutional neural network. The experiment shows that the overall reduction of memory requirement by a factor of 515 and a reduction of operations by a factor of 2.1 does not affect performance.

3.1.1 Model Architecture

Most convolutional neural network architectures, by default, are not designed to run on low-power embedded devices, thus a careful design in terms of structure and learning algorithms must be chosen. To explain the different steps which are necessary to detect an acoustic event, the detection system is divided into three major components, which are illustrated in Figure 3.2. First, the raw time-domain audio waveform is transformed by a frontend into a time-frequency representation. Then the systems extracts features from this representation using an encoder. In a final step the features are mapped to a class vector using a decoder. Note that the separation into encoder and decoder is made for the sake of explanation. Since the final network is optimized end-to-end, both components will contribute to the audio classification task.

In the following, for each step the best option is chosen with respect to memory and computational requirements.

Frontend

The frontend is used to transform the raw audio signal into a time-frequency representation from which features in both, frequency- and time-domain, can

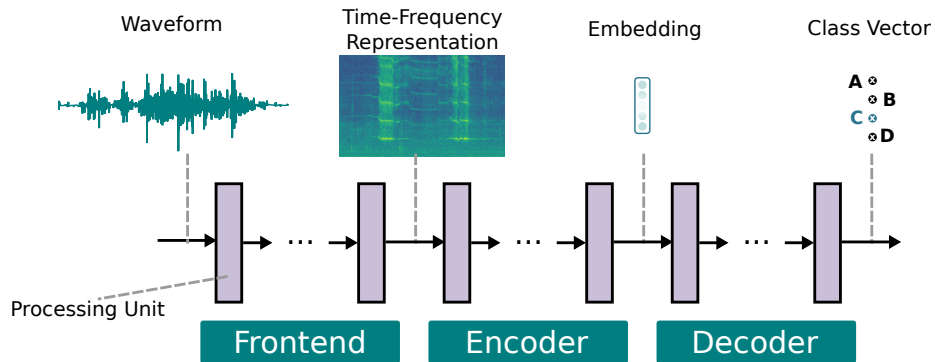


Figure 3.2 Processing flow for the audio classifier. Figure reproduced from Chapter 1.

be extracted. In general these transforms are based on the Short-Time Fourier Transform (STFT), which can be efficiently implemented via the Fast Fourier Transformation (FFT). This fact makes it favorable compared to novel techniques [DS14], which show good results by learning filterbank coefficients from the audio data, but have the major drawback of employing large filterbanks. Mel-scaling, which mirrors the human auditory system, is often used [CHHV15] as addition to the STFT in order to compensate for its linear frequency scale. These additional processing steps may also reduce the amount of data that needs to be processed in later stages of the processing pipeline, which is important since the number of MAC operations of a CNN is directly related to the size of the input field.

As a consequence, in this work a mel-spectrogram has been chosen as frontend to satisfy the real-time constraint. It is calculated with a window size of 32 ms and a hop size of 10 ms using a hamming window. The number of mel coefficients is 64. From the spectrogram multiple frames consisting of 400 vectors are extracted. These frames are input to the feature extractor, thus the network analyzes a time span of 4 s for each frame.

Encoder

The encoder uses a CNN to learn the features of the time-frequency representation. Most CNNs are built from very few basic building blocks: convolution, activation and pooling layers. The concatenation of these blocks introduces a higher depth to the network which has been shown to enhance accuracy [SZ14]. A higher depth results in a higher number of required operations and a higher parameter count. In this work the feature extraction block is therefore limited to two sections consisting of two convolutional layers each, which provides enough parameters to learn significant features but is still moderate in parameter count. Moreover, one convolutional layer with a higher filter size (e.g. 5x5) can be reduced with layers using 3x3 filters, which reduces the number parameters and potentially even improves the classification

		CNN-FC		CNN-C		CNN-CNP				
		Layer type	# param.	# MAC	Layer type	# param.	# MAC	Layer type	# param.	# MAC
Encoder		input	0	0	input	0	0	input	0	0
		frontend	25.6 k	12.7 M	frontend	25.6 k	12.7 M	frontend	25.6 k	12.7 M
		conv 3, 1, 64	1.8 k	32.3 M	conv 3, 1, 64	640	14.8 M	conv 3, 1, 64	640	14.8 M
		conv 3, 1, 64	36.9 k	656.9 M	conv 3, 1, 64	36.9 k	943.7 M	conv 3, 2, 64	36.9 k	236.0 M
		max pool 1x2	0	0	max pool 2x2	0	0	-	-	-
		conv 3, 1, 128	73.9 k	581.0 M	conv 3, 1, 128	73.9 k	471.9 M	conv 3, 1, 128	73.9 k	471.9 M
		conv 3, 1, 128	147.6 k	1040.5 M	conv 3, 1, 128	147.6 k	943.7 M	conv 3, 2, 128	147.6 k	236.0 M
		max pool 2x2	0	0	max pool 2x2	0	0	-	-	-
Decoder		fc 1024	231.2 M	231.2 M	conv 3, 1, 128	147.6 k	236.0 M	conv 3, 1, 128	147.6 k	236.0 M
		fc 1024	1.1 M	1.1 M	conv 1, 1, 128	16.5 k	26.2 M	conv 1, 1, 128	16.5 k	26.2 M
		fc 28	28.7 k	28.7 k	conv 1, 1, 28	3.6 k	5.7 M	conv 1, 1, 28	3.6 k	5.7 M
		-	-	-	avg pool	0	0	avg pool	0	0
		activation	0	0	activation	0	0	activation	0	0
Total:		233 M	2555 M	Total:	452 k	2655 M	Total:	452 k	1239 M	

Table 3.1 Structure, number of parameters and number of required MAC operations for three CNNs. The first (CNN-FC) uses fully-connected layers as decoder, the second (CNN-C) uses convolutional layers as decoder, the third (CNN-CNP) uses convolutional layers as decoder but no max pooling layers. Convolutional layers are defined as *conv filter_size, stride, number_of_filters*. Fully-connected layers as *fc output_dimensions*. Max pooling layers as *max pool pool_size*

performance [SZ14]. In most CNNs max pooling is used to regularize the network, but it has been shown that for small scale datasets the removal of the max pooling layer does not affect the performance [SDBR14]. As a consequence max pooling is removed from the network and the stride of the preceding layer is increased by 1, which divides the required MAC operations for this layer by four while maintaining the same network structure in principle.

Decoder

Table 3.1 illustrates the structure of a network with fully-connected dense layers as decoder (CNN-FC) and a network using only convolutional layers as decoder (CNN-C). For both networks the number of parameters and the number of required MAC operations are listed. It becomes obvious that the fully-connected layers have the biggest impact on parameter count which makes the preceding convolutional layers almost negligible. It has been shown that a fully-connected layer can be replaced by a 1×1 convolutional layer [LCY14], which reduces the number of parameters, and thus the memory footprint, from 233×10^6 to 452×10^3 . The reduction of parameters also further regularizes the network, which is an advantage for training the network. As a last layer average pooling reduces the output of the last convolutional layer to a vector, which size matches the number of labels. The replacement of the fully-connected layers slightly increases the number of MAC operations. However, this increase is acceptable due to the strong memory savings and it can be levelled out by the other previously proposed optimizations.

3.1.2 Final design

The proposed design is denoted as CNN-CNP and is illustrated in Table 3.1. After applying each optimization step as described above the parameter count is decreased considerably by a factor of 515 to 452 k parameters and the number of MAC operations by a factor of 2.1 to 1239 M MAC. Moreover, after optimization the network consists only of convolutional layers. This unified architecture is beneficial for hardware implementation and for the use of convolutional hardware accelerators.

3.1.3 Experiment

After applying these fundamental changes to the network and substantially reducing the number of parameters and arithmetic operations, it needs to be validated that the classification performance is maintained and kept on an acceptable level. For this purpose the two networks (CNN-C and CNN-CNP) have been implemented using Keras [Cho15]. These networks are compared against the best performing implementations from [TGPG16], which are referred to as A and B. The network A has the same structure as the CNN-FC network

Model	A, [TGPG16]	B, [TGPG16]	CNN-C	CNN-CNP
Accuracy w/ aug	91.7 %	92.8 %	-	-
Accuracy w/o aug	77.9 %	80.3 %	86.0 %	85.1 %
# params	233 M	257 M	452 k	452 k
# MACS	2543 M	3533 M	2655 M	1239 M

Table 3.2 Accuracy (with and without data augmentation), parameter count and number of operations for the proposed networks CNN-C and CNN-CNP compared to the top scoring implementations A and B from [TGPG16].

from Table 3.1, the B network is a more complex network with a higher depth and bigger fully-connected layers.

3.1.4 Dataset

The dataset [TGPG16] contains various sound files collected from freely available online sources. It consists of 28 different event types of variable length, e.g. airplanes, violins, birds or footsteps. The total length of all 5223 audio files is 768.4 minutes. The data is split into training and test set. The training set contains 75 % of the original data and is further subdivided into training and validation set with a ratio of 0.25. Although the dataset is strongly biased no data augmentation was performed in the following experiments, since the main focus is on the comparison of the algorithm in terms of structure, resources and classification performance, and not on the augmentation technique. Both networks were trained by minimizing the cross-entropy loss using the gradient-based optimizer ADAM [KB14] with mini-batches of size 128. The optimizers' parameters were left at its default values presented in [KB14]. Testing was done by predicting the probabilities for each class on a 4 seconds window randomly extracted from the test set. The class with the highest probability was chosen as the correct class and compared against the ground truth.

3.1.5 Results

The experimental results are listed in Table 3.2. The values for accuracy of network A and B are taken from the original publication, the values for the number of parameters and number of MACs are calculated based on the information taken from the original publication. The first line of Table 3.2 shows the accuracy results for networks A and B with data augmentation which are 91.7 % and 92.8 %, respectively. As expected, these are better than the corresponding accuracy results without complex data augmentation which are 77.9 % and 80.3 %, respectively. The networks CNN-C and CNN-CNP have an accuracy of 86 % and 85.1 %, respectively. Thus, without sophisticated

data augmentation both proposed networks perform better than the reference network A and even better than the more complex network B.

The analysis of parameter count shows that when 16 bit parameters are assumed, the total memory consumption for the CNN-FC network's weights is 466 MB, which is not feasible for most edge computing applications considering the fierce power and memory constraints for distributed sensing devices. In contrast, the weight storage for the CNN-C and CNN-CNP is approx. 904 kB. Even when considering additional overhead by the implementation on low-power devices such as the NXP KV5x or ST STM32F7 [NXP16, STM16], their flash memory of up to 2 MB is still sufficient to store the parameters of the presented acoustic event detection algorithm.

Considering that the devices mentioned above have a processing capability higher than 430 M MAC/s and processing the input buffer is only required every 4 seconds, they are able to handle the 1239 M MACs of CNN-CNP in less than 4 seconds. Thus the classification can be considered real-time. The evaluation shows that the design specifications could be reduced considerably without impact on performance.

3.1.6 Conclusion

In this section, an acoustic event detection algorithm was presented that exploits the advantages of CNNs while being implementable on low-power microcontrollers. A convolutional neural network has been optimized to be efficiently implemented on resource-limited low-power embedded devices. It was demonstrated that it is possible to reduce the memory requirement by a factor of 515 and the number of operations by a factor of 2.1 without loss of classification accuracy. Due to the flexible applicability of convolutional neural networks it can function as basis for on-device seismic event classifier, which we will present and further optimize in the next section.

3.2 Event-Triggered Natural Hazard Monitoring and Classification on Edge Devices

In this section, we present a scenario where it is mandatory to perform complex decision making on the edge of a distributed information processing system and show based on a case-study how the approach can be embedded into a wireless sensor network architecture and what performance can be expected. We adopt the optimized classifier architecture of the previous section for seismic event classification and further improve its computational performance.

3.2.1 Problem Setting

Natural disasters happen infrequently and for mitigation efforts fast reaction times relative to these rarely occurring events are important, especially where critical infrastructure or even human casualties are at stake [GAC05]. In alpine regions where human habitats including settlements and infrastructure are threatened by rockfalls and other gravity-driven slope failures, wireless sensor networks can act as natural warning systems [IGM⁺12]. They have the flexibility to be deployed in locations that are logistically difficult or dangerous to access, for example an active rockfall scarp. Therefore it is important that these systems run autonomously for long periods of time [GBG⁺12, BGH⁺09b]. Unfortunately, in many cases the close proximity of warning systems to the human habitat has negative implications as noise originating from infrastructure or anthropogenic activities may impact the capabilities and accuracy of a warning system and therefore must be accounted for. In this paper we demonstrate how human noise can be classified, quantified and removed from microseismic signals using an implementation of a convolutional neural network optimized for embedded devices.

Traditionally, continuous, high-resolution data acquisition is used to monitor microseismic signals emanating from structural fatigue [AGS05, OCA⁺12]. These methods are powerful in capturing natural hazard with respect to process understanding as well as hazard warning. However, they suffer severely that in periods of no or little activity continuous high-rate signal amplification and sampling does not provide an information gain while still consuming energy. In addition, these methods scale unfavorably due to the large amount of data produced [WLJ⁺06]. Overall increasing the number of sensor nodes leads to improved detection capability but is clearly limited by the available transmission bandwidth if data is to be processed centrally. One way to reduce the network utilization is to make the sensors themselves more intelligent by shifting the knowledge generation process from the centralized backend closer to the signal sources, e.g. [GBG⁺12]. A novel approach based on a coupled fibre-bundle model exists [FOR16]. It registers precursory patterns of catastrophic events with the help of many threshold triggered sensors and a reduced set of explanatory variables and has recently been tested in a pre-study [FFBV18].

Thus, a system that is optimized to calculate explanatory variables directly on the sensor reduces the logging and transmission cost and thus allows to react with high reactivity on the detection of catastrophic events [BBF⁺11a].

But such an extreme reduction in information content comes at a price: false positives and the inability to characterize events further. While the first has an impact on correct analysis and network performance metrics, the latter is of importance to react adequately on the detection of a disaster. For example if humans are present in a hazard zone they should be warned and a search and rescue mission should be dispatched immediately. Correct and timely information here is of utmost importance to maximize success and avoid expensive interventions on false alarms. Under the constraints given, on-device classification provides a mean to identify humans on-location without the requirement to transmit all sensory data through the network.

We present two aspects: (i) a system architecture for natural hazard monitoring using seismic sensors (geophones) that is designed for detection of precursory rockfall patterns based on the theory of co-detection and (ii) a concept for accurate on-sensor classification of event-based seismic signals to reduce false positives and enhance information by identifying humans in the signal using machine learning techniques.

We evaluate our network and system architecture in two scenarios. In the first scenario we present an outdoor, wide-area sensing system presently deployed in a high alpine natural hazard environment at the Matterhorn Hörnligrat field site at 3500 m a.s.l., Zermatt, Switzerland. We demonstrate the functionality of our system architecture and evaluate its longevity when using event-based data acquisition. The second scenario is a laboratory experiment using an openly available microseismic dataset [MWB⁺18] to demonstrate the feasibility of on-device classification of mountaineers using convolutional neural networks within the wireless sensor architecture. We focus on advanced methods to reduce the memory requirements and latency of an embedded convolutional neural network classifier.

In the following sections we claim the following contributions:

- A variant of a seismic event classifier is implemented on low-power embedded devices using network quantization.
- A sophisticated buffering concept for pipelined inference of a convolutional neural network to relax memory requirements and to decrease latency.
- A realization of a wireless, event-triggered single-channel microseismic sensor system featuring low-power consumption, fast wake-up time and on-device signal processing and characterization capabilities. The system is realized using the Dual-Processor Platform (DPP) hardware design template [SZDF⁺15] and a slightly adapted version of the open-source protocol implementation of the event-based Low-Power Wireless Bus (eLWB) [SDFG⁺17].
- The system was deployed in a field study running on the rockfall scarp of the Matterhorn Hörnligrat at 3500 m a.s.l. from 08/2018 until 02/2020.

3.2.2 Related Work

Rockfall Detection using Seismic Sensors: Seismic precursory patterns before rockfalls have been investigated for several field sites [SDL09, SMG⁺07, AGS05]. These studies are based on microseismic measurements with a portable data logger. Wireless sensor networks have been introduced to cover a larger area while removing the requirement of data retrieval [WJR⁺05, CCVB18, OCA⁺12]. They either provide the option for remote data download or transmit short, event-triggered segments. Unlike in our study, event triggering is done in the digital domain which means that the acquisition system is constantly on.

Acoustic Event Detection: Artificial neural networks have been applied to acoustic event classification [TGPG16, HCE⁺16, CHHV15, CPH⁺17] which includes among others footstep detection. Also footstep detection and person identification using geophones has been studied before [AMK18, PWQ⁺15, LMP⁺16], however only in experiments in a controlled environment, not on embedded devices or using additional structural information. Artificial neural networks have been recently applied to seismic event detection [PGF18]. Especially convolutional neural networks have achieved good accuracies [PGD18, MWBT19b].

Artificial Neural Networks on Embedded Devices: Many studies focus on additional accelerators [ACRB18, HYA⁺18, CKES17] for convolutional neural networks. This approach requires dedicated hardware. Studies on mobile platforms [WLW⁺16] and wearables [MLB⁺17] exist but require a more powerful hardware architecture. A prominent work for low-power embedded devices focuses on keyword spotting [ZSLC17] on a slightly more powerful Cortex-M7 than used in our study. A theoretic strategy for low-memory convolutional neural networks as been proposed in [BB18] which focuses on an incremental depth-first processing idea that resembles our approach. However, they neither focus on sequential data nor on the implementation with a specific buffering system.

3.2.3 System Design

Figure 3.3 illustrates the overall concept. A wireless sensor network consisting of multiple microseismic sensor nodes is deployed in an area where rockfall occurs. The system can be partitioned into sensor nodes that are only used as rockfall detectors (light blue) and sensor nodes that additionally can classify footsteps (dark blue). The two node types have different requirements which will be briefly outlined in the following.

3.2.3.1 Rockfall Detection by Co-detection of Seismic Events

The following describes the principle of detecting precursors of rockfall patterns [FOR16] with threshold-triggered geophone sensors. Multiple

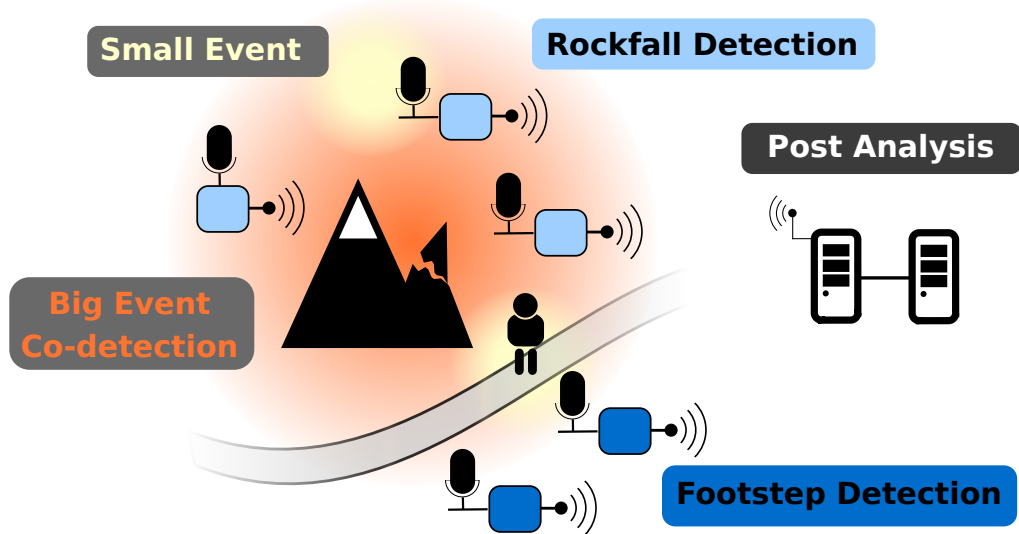


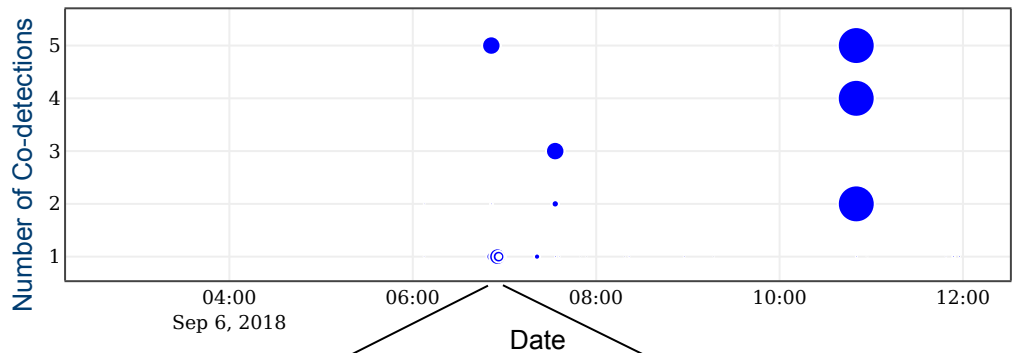
Figure 3.3 Conceptual illustration of a wireless sensor network for natural hazard monitoring based on the principle of co-detection [FOR16]. Multiple seismic sensors are deployed in a hazardous area. The sensor nodes feature the same hardware, a microseismic sensor (geophone), processing, storage and wireless communication subsystems and are able to detect and classify events based on threshold-triggered input signal. If a sensor detects an event it can determine if it originates from a human in the hazard zone or not and possibly trigger an alert. If not it only sends the event information through a wireless low-power network. A basestation collects the information from all sensors for centralized data gathering and further analysis using post-processing methods. Temporal correlation of detected events, e.g. when multiple sensors register an event within the same time window, make it possible to identify the precursors of large mass failures based on the theory of co-detection [FOR16].

geophones are deployed on the rock surface as illustrated in Figure 3.3. If rockfall stimulates a seismic event either due to fracturing/detaching or due to impact, different sensors may register the emerging signal depending on their location relative to the event source [WFM⁺18]. A high amplitude input signal registered at a single sensor can have two causes: Either a large event occurred at distance or a small event occurred in close proximity to the sensor. A co-detection exists if multiple sensors register an event quasi-simultaneously, which allows to distinguish between the two aforementioned possibilities. Furthermore, as lab experiments have shown [FOR16], consecutive co-detections of events can be used to identify rockfall precursors and thus facilitates natural hazard early warning capabilities. Fundamental for this principle is the requirement of many sensors to perform co-detection as well as to cover a large enough area with a sensor cluster [FFBV18]. The data acquisition can be reduced to recording the exact timestamp when the signal exceeds a certain threshold, i.e. capturing events only. While this detection can be implemented very efficiently in hardware using an analog comparator circuit further analysis using signal

processing techniques require a digitizer and processing unit that is typically put to sleep when not in use. A predictable system behavior and a precise time synchronization between all system components and all nodes is important to be able to put co-detected events into context and quantify the underlying processes. A similar system based on much higher frequency acoustic emission signals has been implemented successfully using the Dual Processor Platform (DPP) architecture [SZDF⁺15] and the event-based Low-Power Wireless Bus (eLWB) [SDFG⁺17]. In this work, we adopt these openly available system components to realize an outdoor, wide-area sensing system [Pas18], evaluate and demonstrate its applicability to perform co-detection of precursory rockfall patterns based on low-frequency microseismic signals.

3.2.3.2 Classification with Time Distributed Processing

The co-detection concept allows to reduce false positives, e.g. caused by anthropogenic activity like humans walking by. However, to identify whether a human is present on-site is impossible by just using the reduced set of information transmitted by the event-triggered sensors, i.e. the timestamps of detected events. Since transmitting the raw sensor data of each detected event in real-time for many sensors is infeasible due to bandwidth and energy limitations an approach using on-device classification is advocated. Here, several challenges need to be addressed. Multiple footstep detectors using geophones have been proposed [AMK18, PWQ⁺15] but have not been shown to distinguish well between footsteps and seismic events [MWBT19b] or require further structural information [LMP⁺16]. Convolutional neural networks have shown to be good signal processing tools for classification of acoustic [HCE⁺16] as well as seismic sources [PGD18]. In contrast to other neural network types, such as MLP or LSTM, several convolutional neural network architectures for the special case of seismic event detection have been explored [PGD18, MWBT19b]. Thus, this work focuses on optimizing and implementing an existing CNN-based classifier with known good performance [MWBT19b] to perform well on embedded devices. On the downside, convolutional neural networks have a high memory demand, high memory access rates and a high processing demand. Typical commercially available low-power embedded devices are equipped with two types of memories, static random-access memory (SRAM) and flash memory. On low-power devices the impact of memory usage on the energy efficiency is significant and space in energy-efficient memory structures (SRAM) is limited. However, the inference of a convolutional neural network requires a significant amount of memory to perform the computations, specifically for storing intermediate results and the network parameters. Non-volatile memory, such as flash memory, is typically used to store the parameters of the neural network but the number of read accesses to this type of memory should be minimized since the energy consumption is typically about 6x as high as reading from SRAM [VBM18]. As a consequence the amount of memory accesses required for loading parameters should be reduced, for example by binarization of the network [MTK17].



2018-09-06T06:48:07 CEST



2018-09-06T06:52:07 CEST

Figure 3.4 An example of a rockfall event during the testing phase of the wireless sensor network at the Matterhorn Hörnligrat field-site [WFM⁺18]. Illustrated is data from the monitoring system and two images (before and after significant rockfall) obtained from a remotely controlled high-resolution camera. The top plot shows how a cluster of sensors (see Figure 3.10) co-detected an event over time. The point size indicates the maximum peak amplitude detected in each co-detection within a 0.5 second time window whereas the vertical axis denotes how many sensors triggered within this window. Marked in pink on the right image are areas impacted by rockfall identified by comparing the two images. The mountaineer visible in the lower left corner of the left image is in the danger zone with a number of significant impacts visible in immediate vicinity (pink). Local reports confirmed that no one was harmed in this specific incident.

However this approach comes with a drop in accuracy of about 10%. In our work we apply incremental network quantization [ZYG⁺17], which does not suffer from a reduced accuracy while reducing the network parameter's memory requirement.

For storing intermediate results SRAM is the most energy-efficient memory. However, the intermediate results of state-of-the-art convolutional neural networks do not fit into SRAM. Additionally, convolutional neural networks suffer from a high latency because of the high number of operations required to perform a classification.

In this work we present a novel method to pipeline the computations which relaxes the memory requirements significantly and allows to compute a convolutional neural network in SRAM only while providing a low latency. In the following we call this concept time distributed processing.

3.2.4 Wireless Sensing Platform

The wireless event-triggered microseismic sensing platform presented in this paper is depicted in Figure 3.5 and consists of one single-axis geophone sensor, an analog triggering circuit, a digitizer circuit, an application processor, a communication processors integrated using the BOLT state-full processor interconnect [SZDF⁺15]. We are using a single-axis, omni-tilt geophone sensor since it can be used over a large range of the inclination angles, a characteristic usually not available on multi-axial geophone sensors that require an accurate and level placement over the whole measurement duration. The geophone signal is conditioned and fed to the analog triggering circuit and digitizer circuit. The analog triggering circuit provides the application processor with an interrupt signal if the geophone signal is higher or lower than a given threshold. The application processor will timestamp the detected event and then enable the digitizer system to sample the geophone signal for a pre-defined duration. The processing system is based on the Dual Processor Platform (DPP) partitioning and decoupling the sensing application and the communication onto dedicated processing resources. The interconnect on DPP is realized using BOLT [SZDF⁺15], an ultra-low power processor state-full interconnect which features bi-directional, asynchronous message transfer and predictable run-time behavior. The communication subsystem is based on an IEEE 802.15.4-compatible transceiver (MSP CC430) running eLWB [SDFG⁺17]. Further details on the design of the system architecture can be found in [Pas18] as well as a pre-study using a wired setup in [FFBV18].

3.2.4.1 Analog Triggering Subsystem

The circuit must be capable of amplifying the geophone sensor (SM-6 14Hz Omni-tilt Geophone, ION Geophysical Corporation) signal and comparing it to a predefined threshold trigger. This part of the system is always active, therefore

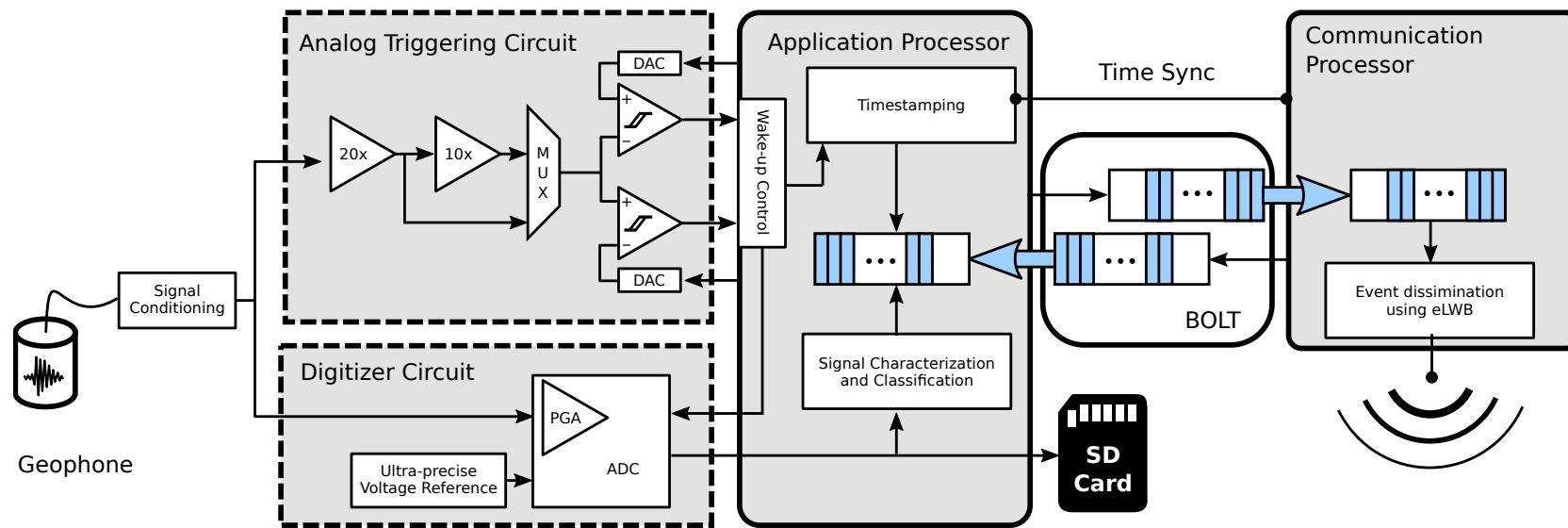


Figure 3.5 System diagram of the event-triggered microseismic sensing platform based on the Dual Processor Platform (DPP) architecture template [SZDF⁺15, SDFG⁺17]. The analog triggering circuit as well as the BOLT interface is always powered whereas all other components can make use of low-power operating modes independently.

it is most sensitive with regards to power consumption. All other system components can be duty-cycled but the trigger must remain powered at all times. An evaluation of different implementation variants showed the superiority of a fully discrete external solution (140.5, 115.2, 22.9 μA respectively) among variants using OPAMP, DAC and comparator circuits internal to the application processor (32-bit ARM-Cortex-M4, STM32L496VG, 80MHz core clock, 3 μA current drain in STOP 2 mode with full RAM retention with RTC on; 1MB flash, 320 kB continuous SRAM), a mix with external and internal components or all-external components respectively. The final design incorporates a dual-sided trigger with individual threshold set-points and a variable amplification (20x and 200x) using MAX5532 12-bit DACs and MAX9019 comparators. The input signal is biased to half the rail voltage and the upper and lower thresholds can be selected between $0 - V_{sys}/2$ and $V_{sys}/2 - V_{sys}$ respectively. Although (in theory) one single-sided trigger should be sufficient, we deliberately chose to implement a dual-trigger system (triggering both on a rising and falling first edge of the seismic signal) to be able to have more degrees of freedom and stronger control over the trigger settings chosen. The overhead for the bipolar trigger system relative to the whole systems power figures in it's different operating modes is negligible (see Section 3.2.7).

3.2.4.2 Digitizer Subsystem

Upon detection of a threshold crossing of the incoming sensor signal the application processor is woken up from an external interrupt and a timestamp of this event is stored. Subsequently the data acquisition system, a 24-bit delta-sigma ADC with high SNR and built-in Programmable Gain Amplifier and low noise, high-precision voltage reference (MAX11214 ADC) is powered on and initialized. It samples the geophone signal at 1 ksps and stores data in SD card storage until the signal remains below the trigger threshold values for a preconfigured duration (post-trigger interval). For this purpose all successive threshold crossings of the sensor signal (the interrupts) are monitored. After ADC sampling has completed the ADC is switched off and all data describing the detected event (event timestamp, positive/negative threshold trigger counts, event duration, peak amplitude, position of peak amplitude) are assembled into a data packet that is queued for transmission over the wireless network along with further health and debug data packets. Using this data, rockfall detection by means of co-detection as described earlier in section 3.2.3.1 can commence using only very lightweight data traffic while the full waveform data is available for further processing and event classification as presented later in section 3.2.5.

3.2.4.3 Wireless Communication System

The communication system is based on the TI CC430 system-on-chip running an adapted version of the event-based Low-Power Wireless Bus (eLWB) [SDFG⁺17]

based on Glossy. This protocol provides low-latency and energy-efficiency for event-triggered data dissemination using interference-based flooding. Since the protocol was specifically designed to be triggered by ultra-low power wake-up circuits it is optimally suited for our application. We use the openly available code¹ with adaptations specific to our platform and the data to be transferred.

3.2.4.4 Application Integration

The Dual Processor Platform (DPP) philosophy using the BOLT state-full processor interconnect [SZDF⁺15] builds on the paradigm of separation of concerns, shielding different system components and run-time functionality from each other for as much as possible. As a side effect this partitioning allows for easy integration and adaption to new applications and/or specifications by allowing to work on communication and application separately. Also, by using well-defined and strongly de-coupled interfaces application re-use is facilitated. In BOLT two queues implemented on non-volatile memory form a strictly asynchronous interface between two processors with guaranteed maximum access times. The obvious drawback of this strict de-coupling however is, that all interaction between the two processors is message based and incurs different end-to-end delays depending on queue fill and access patterns. Therefore tight time synchronization is not readily available. For this purpose a dedicated sync signal is routed between interrupt capable IOs of the two processors. In this way both the decoupling of the two application contexts for sensing and communication as well as tight time sync for accurate timestamping of the detected events based on the network-wide high-precision time sync of eLWB can be achieved [Pas18].

Apart from the event-triggered geophone sensing platform design and its system integration the main contribution of this work is to demonstrate and evaluate a blueprint method for on-board characterization and classification of detected events using neural networks and machine learning techniques. The key techniques and challenges encountered are discussed in the following two sections.

3.2.5 Event Classification

The training of an event classifier for a new field site is always affected by the cold-start problem: Little knowledge about the data is available at the time of initial deployment but this knowledge is required to train a classifier. Moreover, the size and diversity of the dataset is critical for training a good classifier requiring a long time period of samples, for example sensor signals observed over multiple seasons or in different weather conditions. To mitigate this issue we perform a preliminary feasibility study on a dataset with similar characteristics to our application scenario. In this way we can assess the

¹<https://github.com/ETHZ-TEC/LWB>

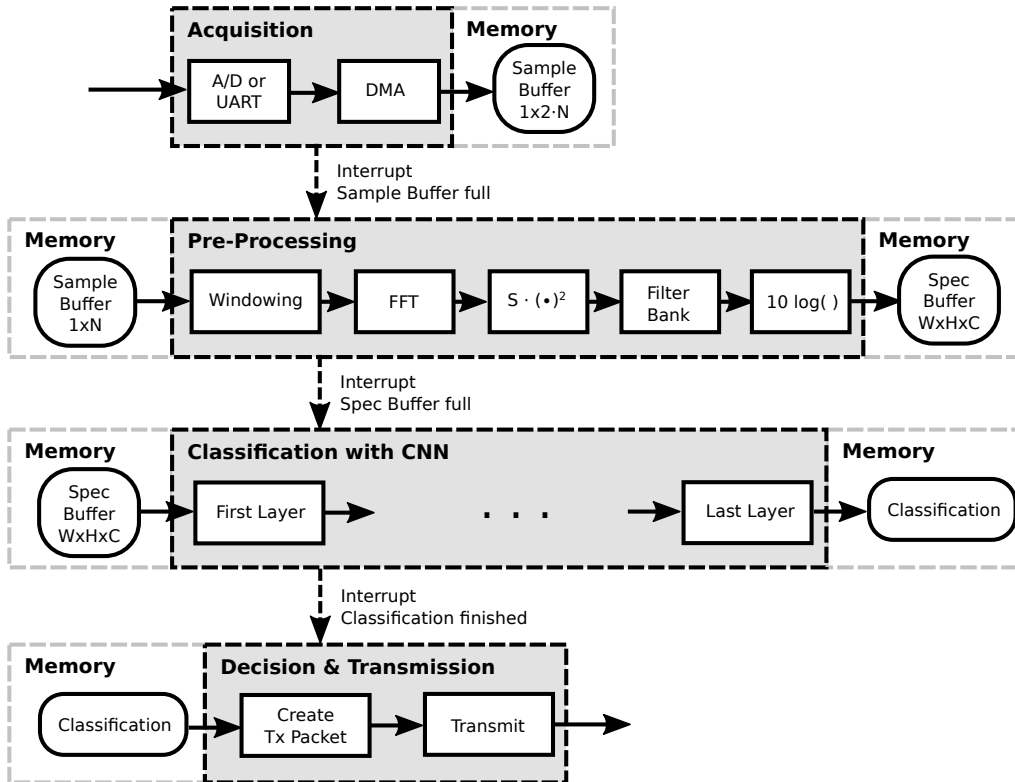


Figure 3.6 Signal processing pipeline for event classification on the experimental platform. The processing pipeline is subdivided into data acquisition, pre-processing, classification with a convolutional neural network (CNN) and decision and transmission. Data sharing between tasks is realized via buffers. Whenever the sample buffer is filled the pre-processing executes and fills one column of the spec buffer. When W columns are filled the classification task is executed. The result of the classification can be transmitted.

energy and storage requirements which are necessary to deploy nodes at a later stage using on-device classification. The neural network can later be retrained easily when an extensive dataset is acquired with the integrated system. The processing pipeline for event classification is illustrated in Figure 3.6 and consists of data acquisition, pre-processing, classification with a convolutional neural network and result transmission. For evaluation purposes we use a development board with the same micro-controller as used for the application processor on the sensor nodes presented earlier in Section 3.2.4. The micro-controller's UART module allows to input data and output results with the flexibility to feed different experimental data for development, debug and performance assessments. In the final design the digitizer frontend with the analog-digital-converter and the network data packet generation feeding result data over BOLT to the communication subsystem will replace these UART modules.

3.2.5.1 Training Dataset

The dataset used is an openly accessible microseismic dataset captured at Matterhorn Hörnligrat [MWB⁺18]. One sample of the dataset consists of a two-minute microseismic recording and a camera image. Both data types coincide in time. The sample's label indicates whether mountaineers are present on the image or not. The dataset also contains additional data structures, such as a list of event timestamps. To be able to use our system model with the dataset two changes have to be applied to the dataset. The seismic sensor used in the reference dataset is a three-axial seismometer (Lennartz LE-3Dlite MkIII). Since our sensor nodes are equipped with a single-axis geophone sensor only a single channel is available as classifier input. Thus, we only use the vertical component of the three-axis dataset for training and testing. The characteristics of the geophone and the seismometer [MWB⁺18] are comparable for the signals of concern in this application. We apply an amplitude triggering algorithm to the two-minute signals and retrieve 12.8 second long event segments to which we assign the same labels as the respective two-minute segments. We set the threshold such that the number of events per two-minute segment is similar in quantity to the event timestamps provided by the dataset. These event segments are then used for training and evaluation. We use the same split for training and test set as defined in the dataset.

The processing pipeline illustrated in Figure 3.6 transforms the digitized geophone signals into a time-frequency representation which the convolutional neural network uses for classification. The dataset samples are acquired over the UART interface to ensure a repeatable experimental setup and subsequently stored in memory using efficient Direct Memory Access (DMA). The samples are transferred via UART using a sampling frequency of 1000 samples per second which is a comparable rate as in the sensing platform presented in Section 3.2.4. When the sample buffer is filled an interrupt triggers the processing task. We perform strided segmentation and segment the signal with a segment size of $N=1024$ and a stride of 512 using a double buffer of size $2N$.

3.2.5.2 Pre-Processing

The pre-processing on the embedded systems is equal to the processing used to train the neural network. It is designed to be efficiently implemented using a Fast Fourier Transform (FFT). Other techniques for audio or seismic classification work directly on the time-domain signal [PGF18], however in that case the convolutional neural network tends to learn a time-frequency representation [SWS⁺15]. By using a FFT the efficiency of its implementation can be exploited in contrast to implementing a filter bank with convolutional filters. The pre-processing task takes the sample buffer as input, multiplies it with a Tukey window ($\alpha = 0.25$) and performs the FFT. The magnitude of the FFT is squared, scaled and transformed using a filterbank. The filterbank maps the FFT bins to 64 bins and thus reduces the data to be processed and stored in

a later stage of the signal processing pipeline. Consecutive log compression creates a distribution of values which is more suitable for the convolutional neural network [CFCS17]. With an input segment size of 12.8 seconds the size of the time-frequency representation is Time x Frequency x Channels (T x F x C) = 24x64x1.

3.2.5.3 Convolutional Neural Network

We use a neural network for classification of mountaineers that is openly accessible [MW18] and which has already been structurally optimized for a reduced parameter set and few computations. It consists of multiple convolutional layers with rectified linear unit (ReLU) activation and zero padding to match input and output size. Moreover, dropout is used to reduce overfitting. In contrast to [MW18] we do not use Batch Normalization layers because we found it to have negligible impact on the test accuracy in our experiment. Our implementation is illustrated in Table 3.3.

For evaluation of the neural network we will use error rate and the F1 score which is defined as

$$\text{F1 score} = \frac{2 \cdot \text{true positive}}{2 \cdot \text{true positive} + \text{false negative} + \text{false positive}} \quad (3.1)$$

To prevent overfitting the neural network is all-convolutional [SDBR14] and dropout [SHK+14] is used. Training is performed using Tensorflow [AAB+15] and Keras [Cho15]. It is accomplished by using 90% of the training set to train while a random 10% of the training set is used for validation and never used during training. The number of epochs is set to 100. For each epoch the F1 Score is calculated on the validation set and the epoch with the best F1 score is selected. The test accuracy is determined independently on the test set.

3.2.5.4 Implementation Challenges on Embedded Devices

To implement the neural network on an embedded device further optimizations are required. The first problem is the storage required for the parameters of the network. The number of parameters is 38,403 which requires 153.6 kB of flash memory using 32-bit values. It is possible to store this amount in flash memory but read accesses to flash should be minimized due to the higher power consumption in comparison to reading from SRAM [VBM18]. We therefore apply Incremental Network Quantization [ZYG+17] which quantizes the parameters to power-of-two values in an iterative weight partition and quantization process. Due to quantization the parameters can be stored as 8 bit integer values and the storage for the parameters of the convolutional neural network is reduced by a factor of 4 without loss in classification accuracy.

The second problem is the size of the intermediate results. The largest

Name	Type	Kernel	Stride	Input size
C_0	Conv2D + ReLU	3x3	1	24 x 64 x 1
C_1	Conv2D + ReLU	3x3	2	24 x 64 x 32
D_0	Dropout	-	-	12 x 32 x 32
C_2	Conv2D + ReLU	3x3	1	12 x 32 x 32
C_3	Conv2D + ReLU	3x3	2	12 x 32 x 32
D_1	Dropout	-	-	6 x 16 x 32
C_4	Conv2D + ReLU	3x3	1	6 x 16 x 32
C_5	Conv2D + ReLU	1x1	1	6 x 16 x 32
D_2	Dropout	-	-	6 x 16 x 32
C_6	Conv2D + ReLU	1x1	1	6 x 16 x 32
A_f	Average (Frequency)	1x16	1	6 x 16 x 1
A_t	Average (Time)	6x1	1	6 x 1 x 1
C_7	Conv2D + Sigmoid	1x1	1	1 x 1 x 1
O	Output	-	-	1

Table 3.3 The structure of the convolutional neural network using 2D convolutional layers (Conv2D) with Rectified Linear Units (ReLU) and dropout layers to reduce overfitting. Number of parameters 38,403.

intermediate result of the convolutional neural network is calculated in layer C_0 . To calculate layer C_1 the output from layer C_0 and additional space to store the output of C_1 is required. With 32-bit values the memory requirement in our case is 245.76 kB which is too large to fit into the SRAM of most micro-controller units. Of course, provisioning this amount of memory would be possible, e.g. using external memory but due to the increase in silicon, access times and energy footprint alternative methods need to be sought for. Since external DRAM is not a suitable solution either we present a method which allows to execute the convolutional neural network using only SRAM and a reduced memory footprint in the following section.

3.2.6 Memory Footprint Reduction Approach: Time Distributed Processing

In this section we present a method to reduce the memory footprint requirement of the convolutional neural network. We will explain this concept with a simple example of a 1D convolutional neural network as illustrated in Figure 3.7. The network consists of two convolutional layers with a 3×1 weight kernel each and strides of 1 and 2, respectively. For illustration purpose we ignore the non-linearity and the bias which are usually part of a convolutional layer. Typically, the network is calculated layer by layer. The input is convolved with the first layer's parameters and the first layer's output is convolved with the second layer's parameters, which requires the intermediate outputs to be simultaneously in memory for the time of execution.

In contrast to this approach we will focus on calculating the output values step by step. Illustrated in red and blue are the respective receptive fields of the second layers' outputs, meaning all values of the input buffer and first layer's output that affect the final output. We first calculate the red output, then we calculate the blue output. This idea is similar to the depth-first approach described in [BB18]. However, additionally we will optimize for the temporal characteristics of our input data with a sophisticated buffering system.

When calculating the red output we already calculated one intermediate result required for the blue output (the point where the local receptive fields in the layer 1 output overlap). If we want to calculate the blue output we see that we only need two values from the first layer's output which have never been calculated before (highlighted in bold). By following the receptive field of these two values we find they depend on four values from the input buffer, among which are two values which have not been used before and two which have been used for calculating the red output. Effectively, this means that we can calculate the output of the network by a combination of new input values, intermediate results and buffered values.

Figure 3.8 illustrates this buffering concept for the same example. We use the following nomenclature: p_i is the number of new input values for layer L_i ; b_i is the number of buffered input values for layer L_i ; p_o and b_o are the respective output values. We call the array of size p_i the processing window of L_i and the array of size b_i the buffer of layer L_i . The value s_i is the stride for layer L_i . The figure presents the last two time steps t_{n-2} and t_n , which results in calculating the blue value in Figure 3.7. Based on Figure 3.7 we set $p_0 = p_1 = 2$, $p_o = 1$ and $b_0 = 2, b_1 = b_o = 1$. We now apply the convolutional layer as before but on the reduced input window. Obviously, we can now shift the input step by step into the buffering system and calculate the output of the convolutional neural network. For each step we acquire two new input samples and discard two older input samples. This makes the buffering system ideal for time series data as is the case in our application scenario. The memory requirement in Figure 3.8 compared to the case in Figure 3.7 is reduced by 35% and the memory requirement for each step is constant. The following section addresses the questions how this general concept is transferred to our application specific convolutional neural network introduced in Section 3.2.5.3 and how the correct size of each layers' buffers is determined.

3.2.6.1 Buffer System Design

Closely examining the inputs of our convolutional neural network in Table 3.3 we see that it consists of 24 timesteps of a 64 value vector. The buffering concept can be expanded to this case if we assume $b_0 \times 64 \times 1$ and $p_0 \times 64 \times 1$ input buffers. Similarly it can be expanded to multiple feature maps (in our case 32) for the intermediate buffers. Before determining the actual buffer sizes we need to determine up to which layer the buffering concept is applicable. The last

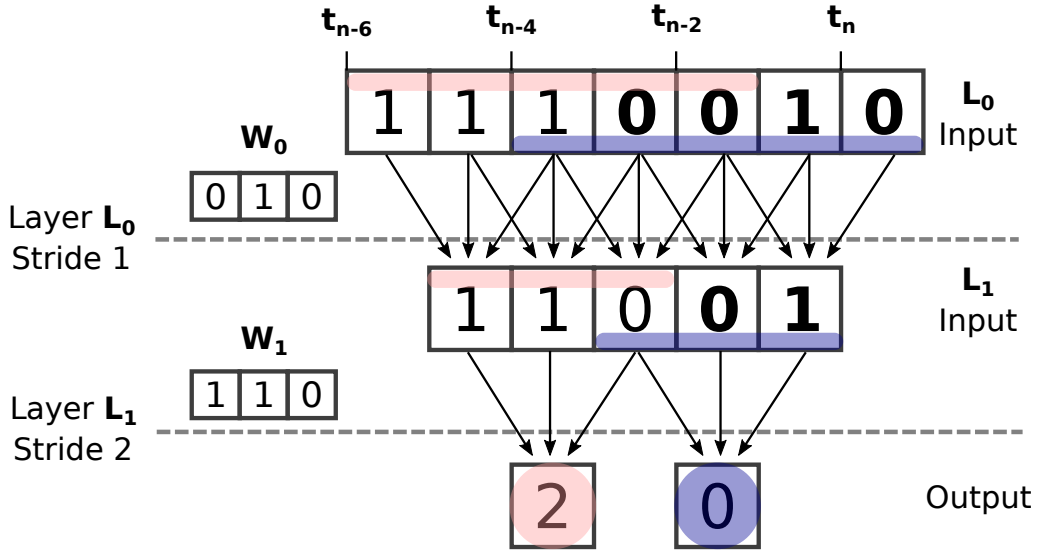


Figure 3.7 The receptive field of the output values is illustrated in red and in blue and grows with the number of layers. The values are calculated layer by layer by convolving the layer input with the respective kernel w .

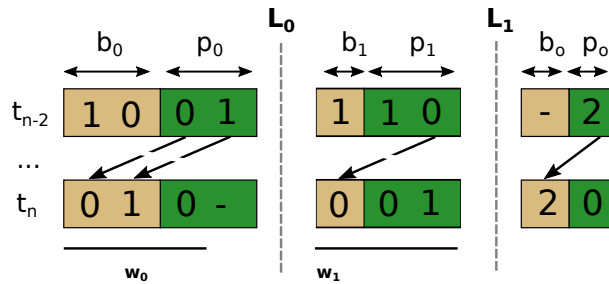


Figure 3.8 Illustration of the buffering concept: Each input of a layer consists of one buffer (yellow) and one processing window (green). Shown are two timesteps t_{n-2} and t_n . Each timestep a new input is placed in the processing buffer of layer L_0 . Subsequently, the layers perform convolutions followed by a left-shift of the processing window by two values.

three layers of the network are average pooling operations and scaling. The output of the time average pooling layer A_t is influenced by every value of the $24 \times 64 \times 1$ network input, which means that after this layer the buffering concept cannot be applied anymore. We can however calculate for each step one of the 6 input values to A_t independently and place them in a buffer. Therefore we can consider all layers up to A_t for the buffer system.

The buffer system is defined by the processing window size p_i and buffer size b_i . The p_i values can be calculated for each layer by considering the convolutional layers up to A_t .

$$p_i = \prod_{l=i}^{L-1} s_l$$

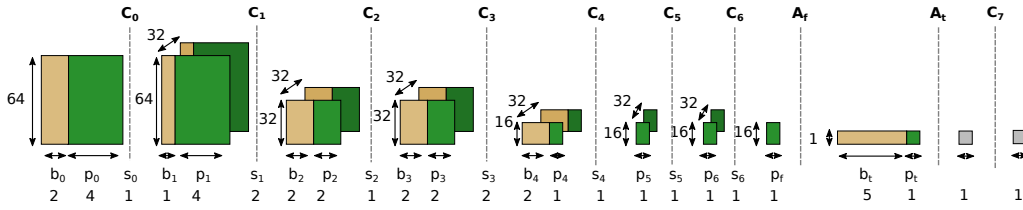


Figure 3.9 Illustrates the buffering architecture for the network from Table 3.3 with input size of $4 \times 64 \times 1$. The input and output of the network are depicted as well as the intermediate results of each layer. C_0 to C_7 are the convolutional layers. A_f and A_t are the average pooling layers for frequency and time, respectively.

where L is the number of convolutional layers up to A_t and s_l is the stride of layer C_l . Note that we ignore the dropout layers for this calculation since they have no influence on inference.

Instead of deriving b_i analytically we followed a systematic approach for each layer. Similar to the approach in the above example we choose the correct b_i based on the layers p_i , p_{i+1} , kernel size and stride.

Finally we can construct our buffering system for the network presented in 3.3. The result is illustrated in 3.9. As discussed before, the number of samples on the frequency axis remains unchanged as well as the number of feature maps.

Using this technique we are able to reduce the memory requirements from 245.76 kB to 85.6 kB for our example application which constitutes a reduction by a factor of ~ 2.87 and as a result allows an efficient implementation on a resource-limited and sufficiently low-power embedded processing device.

3.2.7 Result and Evaluation

In the following we will present our findings, evaluate our system design and demonstrate the advantages of time distributed processing. The dataset [MWB⁺18] is used to assess the performance of the mountaineer classifier. These results are then used in combination with performance data from our field deployment to estimate the lifetime of a sensor node equipped with the event classifier integrated onto the platform's embedded application processor. We do not provide a qualitative evaluation of the rockfall detection system since a labelled dataset including every rockfall during a substantial monitoring period would be required but currently something like this does not exist (worldwide).

3.2.7.1 Field Site Experiments

Nine geophone nodes were deployed in steep, fractured bedrock permafrost on the Matterhorn Hörnligrat field site [WFM⁺18], a site prone to frequent rockfall hazards to evaluate the system characteristics and the suitability for co-detection of rockfall events. The locations of the event-triggered sensor nodes are depicted in Figure 3.10. The system is located right around a frequently

used climbing route and continuously operating since mid-August 2018. The data from the site is fed into a web-based data portal and openly available².

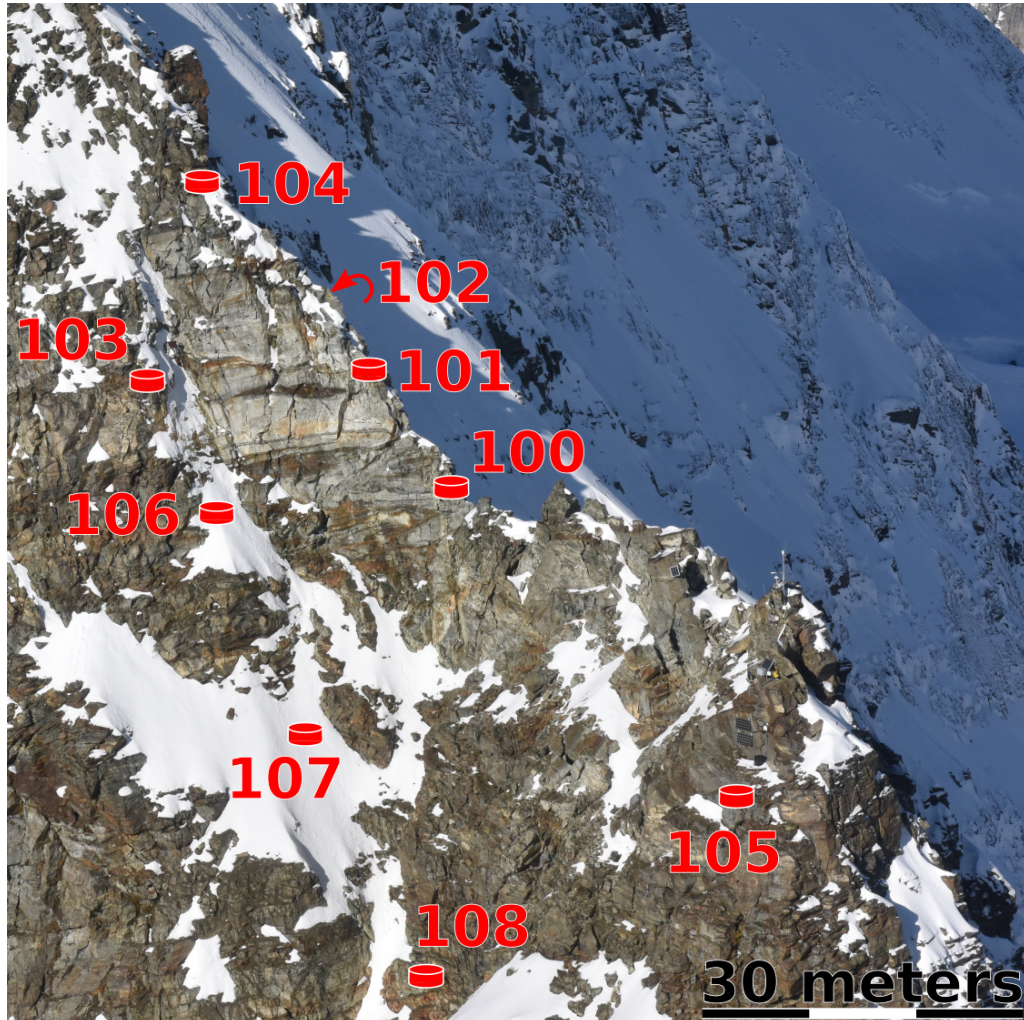


Figure 3.10 Matterhorn Hörnligrat Field site overview: Shown are the locations of the nine wireless geophone sensor nodes. The base station to collect data and transmit it to a data backend is located behind the little tower visible at the upper lefthand corner of the image. The detachment scarp visible is located above a frequently used climbing route.

3.2.7.2 Wireless Sensing Platform Evaluation

The system characteristics of the wireless sensing platform presented in Section 3.2.4 is evaluated in terms of responsiveness and energy efficiency. Lab measurements (see 3.4) have shown that the 20x gain single amplification stage requires only $29 \mu\text{A}@3.0\text{V}$ with the 200x gain dual amplification stage consuming $49.9 \mu\text{A}@3.0\text{V}$. In combination with the other components of the

²<http://data.permasense.ch/>

Measurements	Value [mA]
Active current CC430/eLWB	28
Sleep current CC430/eLWB	0.005
Active current for geophone sensor, frontend and application processor	35
Sleep current geophone	0.035

Table 3.4 Lab measurements of the power performance of the event-triggered microseismic sensor platform at different characteristic operating modes.

trigger frontend that are continuously running the system requires a current of $35 \mu\text{A}@3.0 \text{ V}$ in sleep mode when using the single stage amplification. The active current with ADC operating and application processor running was measured to be 35 mA.

The wake-up time based on an event trigger is important for the acquisition of event waveform data. Since the data acquisition on the ADC is not running continuously in order to save power, no pre-trigger samples are available. Moreover, the delay between threshold-based triggering and the ADC acquiring a first sample will result in data loss with respect to the event signal acquired. We measured a wake-up time from sleep mode of the processor to the acquisition of the first sample on the ADC of only 2.62 ms, which means that on average we lose approximately 3 ADC samples when using a sampling rate of 1 ksp. For most seismic data acquisition systems 1 ksp can be assumed as being a very high sampling rate with a typical value being only 250 sps. Therefore we conclude that using our system architecture this delay is not of significance for the given application.

Time synchronization is a crucial design criteria for our triggered sensing application and especially for using co-detection. Implementing this application using the eLWB protocol based on a synchronous network operating paradigm and Glossy flooding on the lower layer, we are able to achieve time synchronization in realistic operating conditions across a network of tens of nodes within 200 μs [SDFBT17].

For further evaluation of the wireless sensor platform we use data from the first 43 days of the testing phase of our deployment. As can be seen from the statistics presented in Table 3.5, the mean number of events per hour is approximately 28. This value takes into account all events from all sensors and shows that the activity in the network is rather low and low-power performance in sleep mode is most important for this specific application. By looking at the histogram of inter-arrival times of all events in Figure 3.11 we can see that most events are occurring in bursts, having small inter-arrival times below 20 seconds. However, the cumulative density indicates that approximately 15% of inter-arrival times are larger than 100 seconds and thus that there are as well long silent periods, which is also indicated by an inter-arrival time mean of

Statistic	Value
Number of Sensors	9
Days in Field	43
Total Sensor On-Time (h)	28.227
Total Number of Events	29040
Mean Number of Events per Hour	28.14
Mean Event Length (s)	3.5
Mean Daily Acquisition Per Sensor (kB)	788
Mean On-Time Per Hour Per Sensor (s)	10.941
Mean Events per Hour Per Sensor	3.127
Sensor duty cycle (%)	0.304
Average current CC430 (mA)	0.845
Average current geophone (mA)	0.141
Average current total (mA)	0.986
Energy per day (mAh)	23.667
Battery capacity (Ah)	13
Estimated lifetime (days)	549

Table 3.5 The first 43 days of the test phase at our field site have been used to collect statistics about the system behavior. These statistics in combination with lab measurements have been used to estimate the average current of a sensor node and its expected lifetime: ~ 1.5 years using a standard D-size lithium battery (SAFT LSH-20).

~ 1044 seconds. This finding supports our choice of providing the system with an event-triggered sensing system since we can save energy during these silent periods.

The activity statistics can further be used to estimate the energy consumption of our system. With an average event length of 3.5 seconds and the event count per hour per sensor we can estimate the duty cycle of the sensor to be 0.304%. Additionally, using the measurements from the lab for sleep current and active current we can calculate the average current of one sensor node to be 0.986 mA. Using a battery of 13 Ah we can estimate the lifetime of one sensor node to be 549 days.

Using the event triggered sensor around 788 kB are recorded on average every day. Continuous sampling with one of our sensors would produce approx. 259 MB of data per day and sensor, which is an increase by a factor of ~ 328 . Similarly, assuming the geophone is continuously on and the communication processor sends packets as before, the estimated lifetime would be reduced

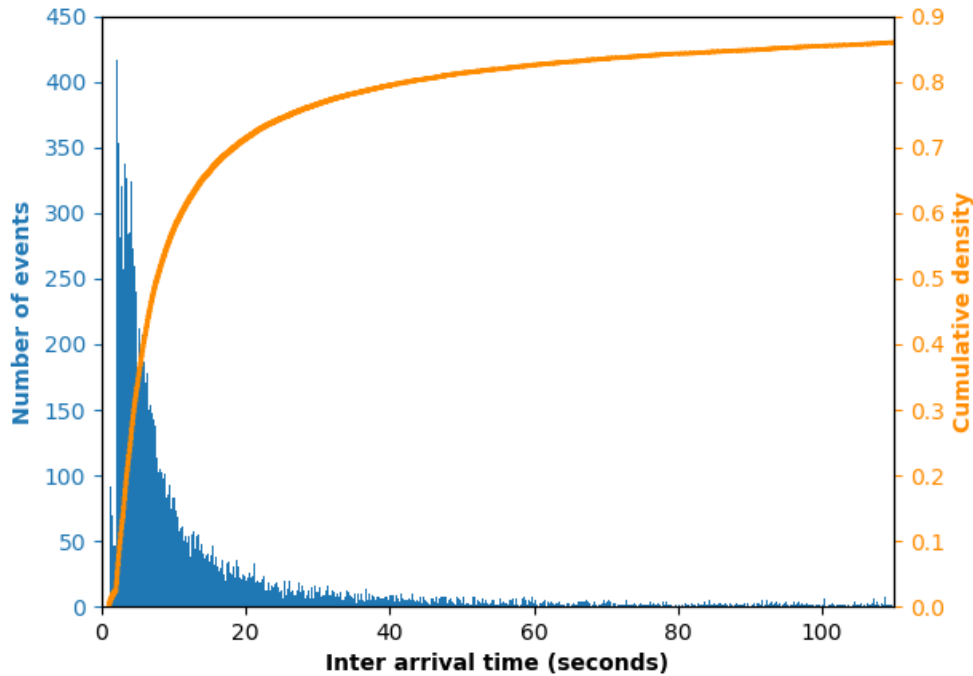


Figure 3.11 The histogram (blue) shows the absolute frequency of the inter-arrival time for 0.1 s bins indicating that a large fraction of events occur in bursts with small inter-arrival times. The cumulative density is visualized in orange.

by $\sim 95\%$ to 15 days excluding effects of higher bandwidth requirements and network congestion that would inevitably occur when building on the same wireless subsystem.

3.2.7.3 Time Distributed Processing Evaluation

The test results for the convolutional neural network from Section 3.2.5 are depicted in Table 3.6. The error rate on the test set for the non-quantized network is 0.0329 and the F1 Score 0.9693. These results are slightly worse than the test error rate after quantization, which is 0.0240 and the F1 score is 0.9779. The effect that quantization improves the test error rate has been also been observed by the authors of the algorithm.

To underline the benefit of time distributed processing we compare the memory requirement and the latency of CNN inference for two scenarios: (i) inference of a 12.8 seconds window (which is the length of the window used for network training) and (ii) inference of an approximately 2 minute long window (the maximum window length we can train with the given dataset).

	Non-INQ	INQ
Top F1 Score	0.9693	0.9779
Top Error rate	0.0329	0.0240

Table 3.6 Results for classifiers evaluated on the test set. The classifier trained with incremental network quantization (INQ) performs better than the classifier trained regularly.

3.2.7.4 Latency

The convolutional neural network must process a time-frequency representation of a specific size, in our case $24 \times 64 \times 1$ values, to perform one classification. The acquisition time is the time it takes to sample and pre-process the data to generate this time-frequency representation. The inference time is the time it takes to perform the calculation of the convolutional neural network. The latency is the sum of the acquisition time and the inference time. Table 3.7 shows the inference time for the two different input lengths. As can be seen, the larger the input the larger the gain of using time distributed processing since the inference time is constant for time distributed processing. Through pipelining the calculation of the convolutional neural network we are more responsive since given the pre-processing details from Section 3.2.5.2 and a p_0 of 4 the acquisition time is 2.56 seconds which is longer than our inference time of 0.44 s.

3.2.7.5 Memory requirement

The memory requirement for the regular layer by layer inference is defined by the biggest intermediate result, since we can reuse the buffers used for calculation. For time distributed processing the memory requirement is defined by the size of the buffers, which are not reused in our implementation. When we compare the two input lengths from before we can see in Table 3.7 that on the one hand we are able to reduce the memory requirement for inference to only 85.6 kB and on the other hand we see that it is independent of the input length. This independence on the input length is another key benefit of time distributed processing.

We used incremental network quantization to reduce our parameters by a factor of 4. The parameters consume 153.612 kB without network quantization and 38.403 kB with network quantization. The sum of weight size and buffer size for intermediate results consumes together around 120 kB which fits in the 320 kB SRAM. Consequently, the parameters can be loaded once into SRAM and the processing can benefit from faster memory access and less energy consumption for reading the parameters.

	w/o TDP	TDP
Inference time (12.8 s window)	1.08 s	0.44 s
Inference time (119.3 s window)	9.55 s	0.44 s
Memory (12.8 s window)	245.76 kB	85.6 kB
Memory (119.3 s window)	2375.68 kB	85.6 kB

Table 3.7 Memory requirement and latency for inference of a convolutional neural network with and without our approach of time distributed processing (TDP). Shown are the values for two different input lengths.

3.2.7.6 Energy analysis

Measurements on our evaluation platform show that the CPU duty cycle during inference is 10%. The measured active current is 15 mA. However, we do not expect an increase in energy consumption when running the neural network on the wireless geophone node since the geophone node does not use any sleep mode during sampling. The application CPU and other components are always-on during sampling. Therefore the major impact is the longer sampling time of 12.8 s plus inference time of 0.44 s. Our estimation for the lifetime of the event-triggered, mountaineer-detection node is 422 days.

3.3 Conclusion



Figure 3.12 Data types, model and optimization used in this chapter. Machine learning-based information extraction from waveform data (seismic and audio) using a data specific frontend (F_W) and convolutional neural networks (CNN) optimized with cross-entropy loss (XE) using a set of labels.

In section 3.2, we have presented a wireless sensor network architecture for the detection of rockfall events using event-triggered microseismic sensors and a method to perform machine-learning-based classification of events on low-power, memory-constraint devices. The system architecture has been designed and optimized for an application in natural hazard warning system providing additional information about human presence in a hazard zone. Our study shows that the lifetime of the system can be significantly extended through optimization for energy-efficiency by using analog triggering and on-device signal characterization. The resulting lifetime is $\sim 37x$ longer than when using continuous sampling while providing the relevant information for rockfall detection by co-detection of seismic events. In this way we demonstrate based

on a real system implementation that information about imminent rockfall and potential hazard to human life including real-time warnings can be acquired in an efficient way, with latencies of only few seconds and in scenarios of realistic scale. Furthermore we demonstrate the performance of this system in a long-term field experiment in a realistic setting on the Matterhorn Hörnligrat, Zermatt, Switzerland.

Moreover, we demonstrate the feasibility of implementing a convolutional neural network for characterization of seismic signals using the example of footstep detection on a low-power microprocessor with a limited SRAM of only 320 kB. Starting from a complex convolutional neural network for audio event classification we apply architectural optimizations (Section 3.1) to develop an architecture for seismic event classification with low memory and computational footprint (Section 3.2). By using network quantization we are able to reduce the parameter's memory requirement by another factor of 4. Additionally, we present a strategy to pipeline a convolutional neural network for temporal data such that we can significantly reduce the inference-time and the inference-memory requirement by a factor of ~ 2.87 and keep them constant independent of the temporal size of the convolutional neural network input.

3.A Appendix

Related Publications

Efficient Convolutional Neural Network For Audio Event Detection

Matthias Meyer, Lukas Cavigelli and Lothar Thiele

arXiv: 1709.09888, Sep. 2017

Contributions: Matthias Meyer developed the concept and discussed it with Lukas Cavigelli and Lothar Thiele. Matthias Meyer developed the code, prepared and performed the experiments and evaluated the results. Matthias Meyer prepared the manuscript with contributions from all co-authors.

 Paper [arXiv:1709.09888 \[cs.CV\]](https://arxiv.org/abs/1709.09888)

Event-Triggered Natural Hazard Monitoring with Convolutional Neural Networks on the Edge

Matthias Meyer, Timo Farei-Campagna, Akos Pasztor, Reto Da Forno, Tonio Gsell, Jérôme Faillettaz, Andreas Vieli, Samuel Weber, Jan Beutel, Lothar Thiele

Proceedings of the 18th ACM/IEEE International Conference on Information Processing in Sensor Networks, Montreal, 2019

Contributions: Matthias Meyer developed the concept and discussed it with Jan Beutel and Lothar Thiele. Matthias Meyer and Timo Farei-Campagna developed the concept for the classifier optimization. They developed the code for the embedded device together with Reto da Forno. Akos Pasztor and Jan Beutel developed the geophone hardware together with Matthias Meyer, Reto da Forno and Tonio Gsell. Matthias Meyer and Jan Beutel installed the geophones, maintained the field site and data together. Matthias Meyer prepared and performed the experiments and evaluated the results with Reto Da Forno and Samuel Weber. Matthias Meyer prepared the manuscript as well as the visualizations with contributions from all co-authors.

 Paper [10.1145/3302506.3310390](https://doi.org/10.1145/3302506.3310390)
 Code https://gitlab.ethz.ch/tec/public/employees/matthias-meyer/2019_ipsn_code/
 Dataset [10.5281/zenodo.1320835](https://doi.org/10.5281/zenodo.1320835)
 Project Webpage matthiasmeyer.xyz/research/ipsn2019/

The following student projects are related to this chapter.

Quantized Convolutional Neural Networks for Embedded Platforms

Timo Pascal Farei-Campagna

Master's thesis, 2018

Event-based Geophone Platform with Co-detection

Akos Pasztor

Master's thesis, 2018

4

Enhancing domain expertise with machine learning and vice versa

In this chapter, we study how the dataset can be made more accessible by combining system context information and deep representation learning. The classifiers presented in the two previous chapters rely on an annotated dataset of high quality. In many scenarios such annotations are not available. Chapter 2 introduced a method to obtain a high quality dataset but only with a significant labor and time overhead. Such repetitive work typically needs to be done by domain experts. Unsupervised learning and semi-supervised learning promises to relieve the expert in these situations. In this chapter, we present two methods how machine learning can support experts: an unsupervised method applied to audio data (Section 4.1) and a semi-supervised applied to a deployment with multiple seismic stations (Section 4.2). Moreover, Section 4.2 highlights the benefits of system context information to enhance machine learning.

Identifying events from a continuously streaming signal source is of interest for many applications including environmental monitoring. In this scenario it can be expected that not all event classes are known and that is is not always certain what distinguishes one class from another. Moreover, real-world datasets collected with sensor networks often contain incomplete and uncertain labels as well as artefacts arising from the system environment. Complete and reliable labeling is often infeasible for large-scale and long-term sensor network deployments due to the labor and time overhead, limited availability of experts and missing ground truth. In addition, if the machine learning method used for analysis is sensitive to certain features of a deployment, labeling and learning needs to be repeated for every new deployment.

Since the Matterhorn dataset is significantly affected by errors as highlighted in Chapter 2, we start to develop representation learning methods for unsupervised tasks using a clean, fully annotated audio dataset. Audio data is highly related to seismic data and the results are relevant to both data types.

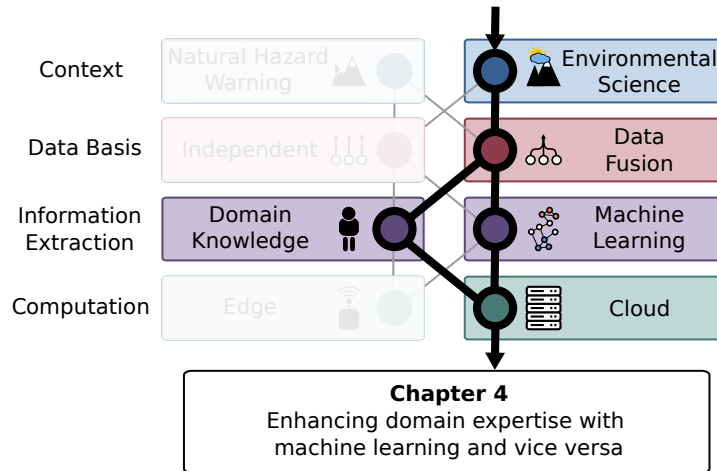


Figure 4.1 This chapter's position in the context of the dissertation

In Section 4.1, we tackle the issue of unknown event classes with an unsupervised feature learning method for audio data exploration. We develop a technique which allows to extract useful information from audio data when annotations are not available. The extracted information can subsequently be used to pre-categorize data and facilitate labeling, clustering or classification tasks. First, we design an audio window predictor based on a Convolutional LSTM autoencoder, which is used to extract features in an unsupervised way. Second, a training method is presented using a novel loss function which optimizes a machine learning algorithm using similarities instead of annotations. The application of the training method leads to distinct features by amplifying event similarities.

To further address the challenges of weak and partially labelled datasets, in Section 4.2, we propose to make use of system context information to enhance machine learning algorithms with additional knowledge. We formalize this knowledge in an information graph and embed it in the learning process via similarity optimization, namely contrastive learning. Based on real-world data we show that this approach leads to an increased accuracy in case of few and weakly labeled data. Moreover, our approach can be used to adapt to new deployments by integrating data from the new location into the training procedure. We find that our approach leads to an increased robustness and transferability of the classifier to new sensor locations.

4.1 Unsupervised Feature Learning for Audio Analysis

The advance of low-power mobile sensors leads to new applications for real-time environmental monitoring including acoustic event detection using wireless sensor networks [DH04]. Acoustic events can be effectively classified using convolutional neural networks (CNN) [EFKN15], if a large labeled training dataset is available [HCE⁺16]. However, for some application scenarios, for example monitoring acoustic emissions in permafrost rock [GBG⁺12], the advantages of CNNs are diminished due to the lack of ground truth. In this scenario distinguishing between relevant events in an audio stream and non-relevant events is non-trivial, since the domain-specific categories of these acoustic events are not known a priori. In this section, we want to develop techniques to explore acoustic datasets for which we do not have access to annotations nor event classes. These techniques can be used to pre-categorize data and subsequently facilitate labeling, clustering or classification tasks. For such tasks it is beneficial that data can be grouped into distinct groups based on their characteristics, which requires a notion of similarity.

Methods based on auto-correlation [BBS08, AB14, YOBB15] are designed to find similarities in the raw data space, but they tend to not perform well for low signal to noise ratios. Other methods, such as autoencoders [HS06], try to extract information from data by transforming the raw data into an intermediate representation. Such methods require a suitable feature extractor which can be learned using data. In contrast to learning using an annotated dataset, unsupervised feature learning provides a method to learn a feature extractor from non-annotated datasets, which has been demonstrated for video [SMS15] and audio [HL09] analysis. Recently, convolutional long short-term memory (ConvLSTM) layers have proven to be effective for time series data [XCW⁺15] and for unsupervised video analysis [FGL16]. ConvLSTM layers have not yet been often used for audio analysis [ZCJ16], despite their advantages for time series data.

In this section we explore the usefulness of ConvLSTM layers by designing an autoencoder-based audio window predictor, which is used to learn representative features of acoustic events. One disadvantage of such an approach is the fact that the objective of an autoencoder is to recreate the original signal but not to compare two signals with each other. Thus, we can not immediately leverage information about event similarities. To circumvent this issue, we propose a novel approach to train an autoencoder with respect to generating and extracting distinct features. The approach is designed to diversify the features based on inter-sample similarities and thus allowing for tighter clustering of similar events. These two contributions are used to develop a system which can infer distinct event features to be used for data exploration tasks such as labeling, clustering or classification.

4.1.1 Approach

An autoencoder can be used to extract information from a signal without the need of a labeled dataset. The autoencoder encodes an arbitrary input signal into a compressed intermediate representation (embedding) from which it learns to recreate the original signal [HS06]. In our scenario the input is taken from audio data which is transformed into a two-dimensional time-frequency representation as is often used in audio analysis [CHHV15]. As illustrated in Figure 4.2 (a), from this time-frequency representation three windows of length T_f are extracted which are collectively used as input of the autoencoder. Instead of learning to predict the same input, the autoencoder displayed in Figure 4.2 (b), learns to predict the next three windows. To achieve this goal, the input windows are transformed with an encoder into an intermediate representation (embedding), which is a matrix $\mathbf{E} \in \mathbb{R}^{C \times T \times F \times W}$ with four dimensions which are related to the input dimensions channel, time, frequency and window. The embedding is then used to predict the next audio window and to compute a similarity loss.

The audio window prediction is learned by re-transforming the embedding into a shape similar to the input using a prediction decoder. Then, the mean squared error loss between decoder output and the ground truth is used to update the parameters of encoder and decoder. Simultaneously, the encoder parameters are updated with a similarity loss. To compute the similarity loss, it is assumed that an autoencoder using convolutional layers can be trained such that each channel of the embedding contains information about a specific feature of acoustic events. To reduce the amount of data, we apply average pooling to the embedding by computing the average along all but the channel dimension. Thus, averaging reduces the embedding matrix \mathbf{E} to a C -dimensional vector $\mathbf{v}_f \in \mathbb{R}^C$ (feature vector). Here, we assume that also the mean of each channel contains relevant information and defines an acoustic feature.

Since we want to optimize our embedding based on similarities, we need a method for comparison. The Kullback-Leibler divergence can be used as a measure how two probability distributions differ. We therefore need to transform our feature vector into a probability distribution, which can be done by transforming it with the softmax function into a categorical distribution (from here on called feature distribution). Now, similarities between two audio samples can be characterized by the statistical distance between their feature distributions, a concept similar to [HK15]. Since training of a neural network is typically done in batches of size N , we propose a pairwise loss function $L(n, m)$ based on similarities:

$$L(n, m) = \begin{cases} \text{KL}(B^{(n)} \parallel B^{(m)}), & \text{if } f_{n,m} < \text{threshold} \\ \max(0, \text{margin} - \text{KL}(B^{(n)} \parallel B^{(m)})), & \text{else} \end{cases} \quad (4.1)$$

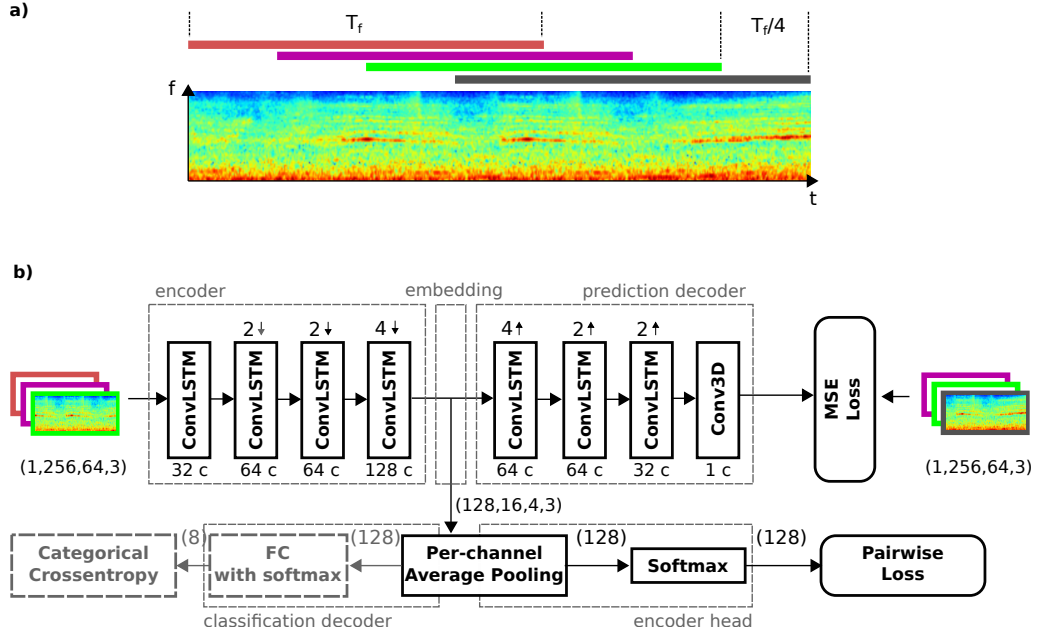


Figure 4.2 (a) Model input, consisting of three windows extracted from a mel-spectrogram [Mer76] with window size T_f and stride $T_f/4$. The fourth window is to be predicted by the model. (b) Model of the audio window predictor, which is trained to predict the next window of the mel-spectrogram input. Multiple ConvLSTM layers are used differing by number of channels (e.g. 32 c) and down-/upsampling (e.g. $2 \downarrow$). Per-channel average pooling reduces the embedding to a lower dimensional feature vector. A batch of feature vectors is used to compute a pairwise loss in an unsupervised way. Labels are only used during evaluation in which a simple classifier, consisting of a fully connected (FC) layer, is used to evaluate the usefulness of the feature vectors for classification tasks.

$$f_{n,m} = \frac{\text{KL}(B^{(n)} \parallel B^{(m)})}{\frac{1}{N^2} \sum_{o=0}^{N-1} \sum_{p=0}^{N-1} \text{KL}(B^{(o)} \parallel B^{(p)})} \quad (4.2)$$

where $B^{(n)}$ defines the feature distribution of the n -th element of the batch. KL is the Kullback-Leibler divergence and the two parameters *threshold* and *margin* can be adjusted based on the actual data. $f_{n,m}$ is a measure for the similarity of two elements of the batch. Therefore thresholding $f_{n,m}$ means that the loss $L(n,m)$ penalizes events based on similarity, for similar events the feature distribution gets more confined while dissimilar events are shifted away from each other. We expect, that this leads to a grouping of features based on similarity in the feature space, which is beneficial for differentiating a pool of events when no labels are available. The overall loss per batch is $\sum_{n=0}^{N-1} \sum_{m=0}^{N-1} L(n,m)$, which can be backpropagated through the network and affects the weights of the encoder. The total loss for the encoder weights is the sum of the autoencoder loss and the pairwise loss. The autoencoder enforces a meaningful representation while $L(n,m)$ amplifies similarities and dissimilarities.

Thus, this method optimizes for inter-sample similarities in contrast to the method from [XGF16], which optimizes for fixed cluster centers.

For the actual implementation an autoencoder is used as an audio window predictor, which is illustrated in Figure 4.2. Data windows of length $T_f = 2.56$ s are extracted from the audio waveform with a hopsize of $T_f/4$ and transformed into a mel-spectrogram. The audio window predictor considers three consecutive windows and predicts the following window. Each ConvLSTM layer uses a 3x3 filter kernel and is followed by a ReLU activation and batch normalization. The last convolutional layer (3D convolution) applies a 3x3x1 kernel with a linear activation function. Keras [Cho15] and Tensorflow [MAP⁺15] are used for the implementation.

The training with batchsize $N = 16$ is done in two steps. First, only the audio window predictor is trained by minimizing the mean squared error (MSE) between the decoder output of the audio window predictor and the true next window, initializing the weights of the whole system. After training the audio window predictor for 6 epochs the system is trained for another 3 epochs by minimizing the above mentioned pairwise loss at the encoder head output and the MSE at the decoder output, simultaneously. In the next section a system trained with this two-step procedure (PairLoss) will be compared against a system trained only by minimizing the MSE at the decoder output for 9 epochs (w/o PairLoss).

4.1.2 Evaluation

An acoustic event dataset from [TGPG16] was used for evaluation. In Figure 4.3 eight categories are listed which were chosen to have an approximate uniformly distributed number of samples. The dataset has been partitioned into training (75%) and test set (25%). The number of embedding channels has been set to $K = 128$ to maintain an accurate prediction and to provide enough features to explain the data.

Table 4.1 Test accuracies for classifier and clustering algorithm. Note: The reference classifier uses supervised training for all weights, the others only for the fully connected layer

Classifier			Clustering	
Reference	w/o PairLoss	w/ PairLoss	w/o PairLoss	w/ PairLoss
(90.85 %)	69.59 %	78.94 %	38.32 %	52.23 %

We will evaluate the usefulness of the feature vectors for classification and clustering applications. Table 4.1 highlights the results for these two evaluation scenarios.

First, it is shown that the feature vectors contain useful information by learning a

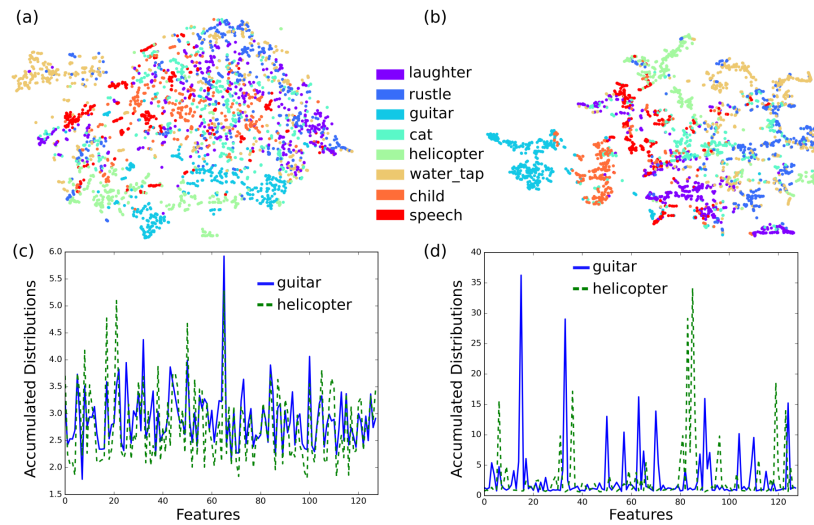


Figure 4.3 t-SNE embedding [vH08] of the feature distributions for a system trained without PairLoss (a) and with PairLoss (b). Accumulated feature distributions for all training samples of the category *guitar* and *helicopter* for a system trained without PairLoss (c) and with PairLoss (d).

mapping from the 128-dimensional feature vectors to the 8 categories. A simple linear classifier (classification decoder), consisting of only one fully connected layer, is trained using the feature vectors as input (Figure 4.2). The classifier is optimized by minimizing the cross-entropy loss between the classifier's output and the ground-truth labels. During evaluation, the weights of the ConvLSTM encoder are fixed, meaning they are not affected by the optimization.

Table 4.1 highlights, that the system pre-trained using PairLoss outperforms the one pre-trained without PairLoss, which can be attributed to the different feature distributions, as illustrated in Figure 4.3 (c) and (d). The plot displays the accumulated feature distributions for two event categories. In contrast to standard training, it becomes apparent that PairLoss training results in two different event categories showing distinct patterns in their accumulated feature distribution. Therefore, only a few elements of the feature vector contain most of the information about an acoustic event. The system does not perform as well as the reference implementation which uses all labels to train the encoder and the classification decoder simultaneously. Moreover, related work [TGP16] has demonstrated a better accuracy of 92.8% on the same dataset. However, they are using a different architecture and supervised training with all available labels. The evaluation in this section is meant to demonstrate the usefulness of the features obtained by training without labels.

The second evaluation demonstrates the clustering performance of our approach. Table 4.1 shows that using a system trained with PairLoss shows better clustering accuracies than a system trained without. For this evaluation k-means clustering (8 clusters) is applied to the t-SNE embedding of the

feature distributions. The best cluster assignment is chosen using the Hungarian algorithm [Kuh55].

4.1.3 Conclusion

In this section, we have designed an unsupervised feature extraction system for audio data consisting of an autoencoder-based audio window predictor. Moreover, we have introduced a novel approach to train such a system by proposing an in-batch similarity optimization technique. We have experimentally shown that applying the new method generates distinct features which increases classification accuracy by 13 % and clustering accuracy by 36 %.

This section has shown that similarity optimization can be beneficial for unsupervised information extraction. These results motivate to further explore the possibilities of similarity optimization. In the next section we will address the issue that real-world datasets often contain incomplete and uncertain labels. Moreover, we tackle the problem that machine learning algorithms are often very sensitive to certain features of a deployment which leads to a decreased transferability of the algorithm do new deployments. We will extend the concept of this section by introducing annotations to similarity-based optimizations resulting in a semi-supervised learning algorithm. We build on progress in the machine learning community and use another concept for similarity optimization, namely contrastive learning, to make models based on weakly-labeled, real-world seismic data more robust to changes in their data distribution.

4.2 Using system context information to complement weakly labeled data

Real-world datasets originating from environmental monitoring deployments often contain artefacts arising from the system environment and contain incomplete and uncertain labels. One example of machine learning applications is natural hazard monitoring for slope failure detection [HOF13, DMH⁺16]. Here, high misclassification requires careful retraining and post-processing [HPM⁺17]. In this setting, comprehensive manual annotations are infeasible for large-scale and long-term sensor network deployments due to the labor and time overhead [MWBT19a]. Moreover, rare occurrences of relevant events makes collecting a comprehensive event dataset difficult and leads to imbalanced datasets.

Hence, the process is error-prone and requires significant domain expertise. However, experts might not be available throughout the whole deployment periods of the sensor network, which inevitably leads to an annotation set containing noisy annotations limited in certainty (weak labels), time and/or subset of sensors (few labels). In addition, as long as the learned features and classifiers are sensitive to the detailed properties of the environment and the sensors, labeling and learning needs to be repeated for every new installation or classifier performance is decreased [WHv⁺21]. Moreover, new installations potentially introduce new event characteristics or error sources which might require new label classes. Therefore, there is a close link between weakly labeled data and classifier robustness with respect to certain environmental variations.

Fortunately, real-world deployments provide additional sources of information which could be beneficial for learning, such as correlation of sensor data due to sensor proximity. However, this information cannot be easily captured by the prevailing data/annotation pairs used for learning. Similarity learning [SKP15, MBT17], such as contrastive learning [HFW⁺20, CKNH20, SGZZ20] allows to establish relations between data pairs. However, their capability to integrate system context information is limited.

To address these challenges, we propose to transfer the concept of knowledge graphs [HBC⁺21] to learning by using it for storing information about data similarity. Moreover, we extend the prevailing data/annotation learning concept to allow any data point to be an annotation for any other data point. This is accomplished by utilizing the following concepts: (i) injecting all available knowledge in form of an information graph and sampling from it, (ii) transforming the data into a common representation and (iii) the use of contrastive learning to train the system. We show that using these concepts to formalize system context information and using the additional knowledge in the learning phase leads to an increased accuracy in case of weakly labeled data and leads to an increased robustness and transferability of the classifier to new sensor locations.

Our main contributions are:

- We present a method which uses system context information to counteract the negative impact of few and weak labels by combining contrastive learning with an information graph.
- We present a unified learning process in which annotations are encoded as Gaussian random vectors to treat them similar to data.
- We demonstrate on a dataset gathered from two real-world deployments in the Swiss alps, how the method can be used to train a classifier with improved generalization performance across sensors with diverging characteristics.

4.2.1 Dataset

Illgraben deployment

In this section, we use data from a real-world deployment of seismic sensors at Illgraben, Switzerland [WHv⁺21]. The sensor array consists of 8 seismometer (ILL01-08), each having three channels, one vertical and two horizontal. The sensors are deployed at distances of hundreds of meters up to several kilometers away from the area of interest. We aim to distinguish seismic signals from 3 different types of events namely earthquakes, slope failures and noise signals. The Illgraben event catalog was created by visual inspection of the vertical channel of the continuous seismic recordings and their spectrograms by experts for a time period between end of May and mid August 2017. The earthquake catalogs provided by the Swiss Seismological Service (SED) and the European-Mediterranean Seismological Center (EMSC) served as additional ground truth for providing correct earthquake labels. The Illgraben event catalog consists of 320 to 560 time segments per station each containing an event, summing up to 32.5 hours of labelled seismic data recorded at a sampling frequency of 100 Hz. Depending on the event duration, the segment duration ranges between 0.1s up to 337s with a median duration of 26s. In addition, the dataset contains randomly sampled, verified time segments without activity with a total duration approximately equal to the event segment's total duration, which is labelled as miscellaneous (*Misc*). Due to the low signal-to-noise ratios (SNRs) of seismic signals generated by smaller mass movements, the data set contains more annotated events on stations close to the detachment area (ILL05-08). We treat this dataset as a multi-class single-label problem: each time segment can only be assigned one specific class. We consider the assumption to be valid, as it is even for an expert nearly impossible to distinguish between two overlapping signal sources.

Matterhorn deployment

In addition to the data from the Illgraben deployment, we use data from the Matterhorn deployment presented in Section 1.3, see also [WBF⁺19]. More

	count	mean	std	min	25%	50%	75%	max
Total	3011	42	36	1	31	40	40	337
Events only	1315	43	53	1	12	26	53	337
Misc only	1696	41	13	40	40	40	40	224

Table 4.2 Statistics describing the event durations of the Illgraben dataset for the subset containing only events, for the subset containing randomly sampled and verified miscellaneous segments (Misc) and the combination of both. Given are mean, standard deviation (std), minimum (min), 25 percentile (25%), median (50%), 75 percentile (75%) and maximum of the event duration. The unit for each column is *seconds*, except for the *count* column which is without unit.

specifically we use data of one seismic sensor and the corresponding annotation set which was developed in Chapter 2. The annotation set is split into two subsets, namely into training (data from 2016) and test set (data from 2017). In Chapter 2, the annotation set has been created by verifying events on a co-located timelapse camera, which takes a high-resolution picture every 4 minutes. Each image is compared to a 2 minute seismic segment centered on the timestamp of the image. Each image and 2 minute segment is annotated jointly by visually identifying every event in each data type. Unlike the Illgraben event catalog, the annotations are made on a per-segment basis and not on a per-event basis. More details can be found in Section 4.2.5 and in Chapter 2.

4.2.2 Related work

Machine learning for mass movement monitoring: First attempts to automatically classify seismic signals of mass movements using machine learning were made by [HOF13, DMH⁺16, HHVH⁺18] using hidden Markov models (HMMs). However, high misclassification required careful retraining and post-processing. As an alternative [HPM⁺17, PHM17] used random forest [Bre01] to automatically classify predicted seismic events. They computed more than 50 features of the detected seismic events in the time and the frequency domain as classifier input. However, this method requires an preceding event detection algorithm with additional parameter tuning. To eliminate this additional step and to optimize for monitoring purposes, [WHv⁺21] adjusted their approach, and classified windowed seismic data on the continuous data stream. In several other domains deep learning models have proven superior to random forest classifiers in terms of accuracy. Therefore, we make use of deep learning classifiers for mass movement classification in this section.

Similarity learning: In-batch similarity optimization has been studied using pairwise loss [HK15, MBT17] and triplet loss [SKP15]. However, these methods could not compete with purely supervised methods for classification problems. More recently contrastive loss [HFW⁺20, CKNH20] was used for similarity optimization. The use of contrastive learning has steadily

increased in the recent past and lead to state-of-the-art results for many datasets in computer vision [BHB19, HFL⁺18, TKI19, GSA⁺20] and in audio analysis [SGZ20, FOM⁺20]. Contrastive learning is also used for out-of-distribution detection [WBR⁺20, TMJS20] which makes it a suitable approach for our application scenario of weakly annotated data. Semi-supervised [IG20] and supervised [KTW⁺20] contrastive learning approaches have been presented using labels to connect similar pairs in the batch. In this section we build upon these concepts and introduce a generic way to implement system context information and annotations.

4.2.3 Method

In our scenario, two major issues need to be addressed, namely (i) few and weak labels and (ii) classifier robustness. The first issue requires an improvement and extension of the annotation set, the latter requires that the learning method can adapt to out-of-distribution samples.

In contrast to real-time classification, environmental monitoring usually relies on post-processing of a long-term dataset. Therefore, an extensive dataset is usually available for training albeit not always thoroughly annotated. In our scenario, we can make use of non-annotated data by using general assumptions about the specific sensor deployment, for example about sensor proximity: The same event is captured by multiple seismometer channels and possibly multiple stations, but with different signal signatures. These differences are caused by groundwave propagation as well as properties of the seismometers, for example ground coupling. Thus, we obtain "different views" of the same event. Contrastive learning has shown to benefit from such different views. Intuitively speaking, contrastive learning achieves an embedding of data samples in a latent space by moving representations of different views of the same event closer together while increasing the distances of representations of different events.

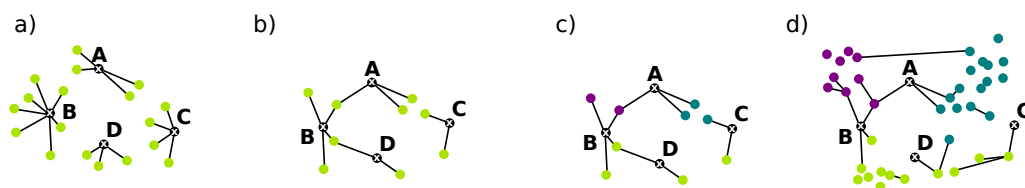


Figure 4.4 Various use cases of the information graph. Each graph contains nodes which represent data of a specific type (colored dots) such as annotations (black dots). Nodes are connected by a line if the associated data is similar. The graphs for single-type data represent the prevailing data/annotations pairs: (a) multi-class, single-label, or (b) multi-class, multi-label. The information graph is suitable to additionally represent multi-label, multi-class, multi-type problems (c) as well as inter-node dependencies (d).

To combine contrastive learning with all available system context information, we propose to make use of an information graph, which holds annotations as well information about the relation between time segments, channels and stations. More generally, our design of the information graph consists of multiple nodes, each representing an item of the dataset. A node can represent for example a segment of seismic data or an annotation. By introducing weighted edges between these nodes, we can establish relations between dataset items. In our design, an edge and its respective weight define the similarity between the two connected nodes. Using this design, a dataset consisting of data and its respective annotations can be reformulated as information graph. This graph contains data nodes and annotation nodes which are connected by an edge if a data item is annotated with a given annotation label as illustrated in Figure 4.4 (a-c). Moreover, we can extend the concept such that a data item is an annotation for another data item by connecting two nodes with an edge if the data item represented by each node is similar, as highlighted in Figure 4.4 (d).

We expect that by using system context information to generate these edges we simultaneously (i) improve our annotation set by adding additional information to each annotated segment and (ii) we can include non-annotated, out-of-distribution samples in the training process.

For our application, the information graph is filled by subdividing the seismic signals into segments using a window length T_w . Each segment is represented by a node in the information graph. An edge is introduced between segments A and B if the segments overlap in time and A is from a different station or different channel than segment B. To reduce the possibility of learned shortcuts [FOM⁺20] no edges to segments of the same channel are added.

To make use of contrastive learning the data must be transformed into a common space. In our approach, the information graph is used to train a model $f(\cdot)$ which embeds each segment x_i into a common space $z_i = f(x_i)$, with $z_i \in \mathbb{R}^d$. Similar to related work, we separate the model into an encoder and encoder head [CKNH20]. As illustrated in Figure 4.5, a fixed number N_e of edges is sampled from the graph for every batch during training and connected data segments are loaded. Any duplicate segments are removed from the batch before computing $f(\cdot)$, leading to a number of data segments in the batch of $N \leq 2N_e$. Each data segment is encoded by the encoder and subsequently transformed by the encoder head into an embedding vector z_i .

By sampling the edges we construct a subgraph of the information graph (called batch graph from here on) with non-negative adjacency matrix $\mathbf{A} \in \mathbb{R}_{\geq 0}^{N \times N}$. Thus, the batch graph is a sparse subset of the total graph consisting of few edges and their respective nodes. Unfortunately, it can occur that two nodes can exist in the batch graph which are connected in the information graph but not in the batch graph, since they were introduced to the batch graph through the sampling of edges not shared by the two nodes. This issue could have a detrimental impact on training because any nodes not directly connected should be considered dissimilar. As a resolution, we make use of the fact that the m th

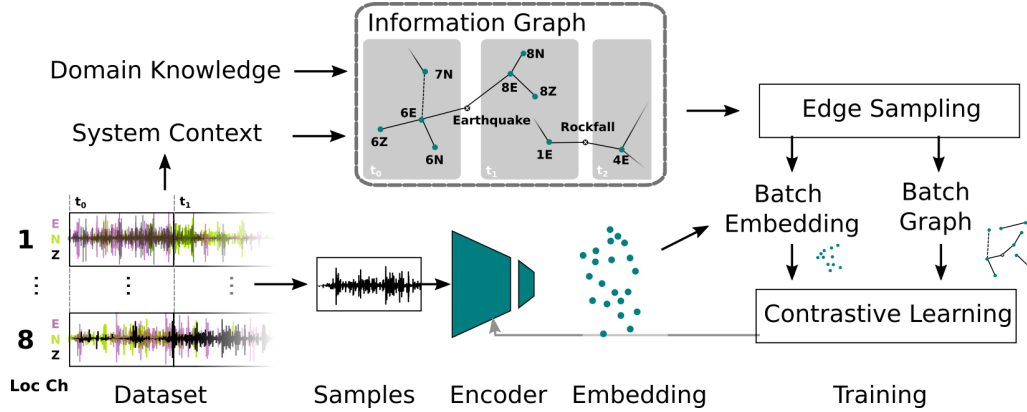


Figure 4.5 The central entity is the information graph which combines knowledge from domain experts, for example annotations or signal propagation behavior, as well as dataset-specific knowledge of each data segment, for example **location**, **channel**, time. The combination of contrastive loss, information graph and encoder allows to learn a suitable embedding for the classification task.

power of an adjacency matrix represents a graph in which m th order neighbors are connected. Thus, if we compute $\mathbf{B} = \sum_{m=1}^M \mathbf{A}^m$, we obtain an adjacency matrix for the batch graph in which all nodes with an direct or indirect link are directly connected. We have found $M = 2$ to be sufficient in our case which includes all second order neighbors. In this setting, we define the contrastive loss between a pair s, t (source and target of an edge), with the adjacency matrix \mathbf{B} as follows:

$$L_{s,t} = -B_{s,t} \log \frac{\exp(\phi(\mathbf{z}_s, \mathbf{z}_t)/\tau)}{\sum_{n=1}^N \mathbb{1}_{[B_{s,n}=0]} \exp(\phi(\mathbf{z}_s, \mathbf{z}_n)/\tau)} \quad (4.3)$$

where $B_{s,t}$ represents the weight of the edge connecting s, t . $\phi(\cdot)$ is a similarity function, which in our implementation is the Cosine similarity. τ is a temperature scaling. The indicator function $\mathbb{1}$ is evaluating to 1 iff $B_{s,n}$ is zero.

The previous steps describe how to transform the seismic segments into the embedding space to compute the loss $L_{s,t}$. Now, we need to transform the annotation information contained in the information graph into a format which can be integrated with the loss function. Annotations are considered in the information graph by introducing N_c anchor nodes, where N_c equals the number of classes. Each segment belonging to a class is connected to the anchor node of that class by an edge. Two strategies can be employed to compute Eq. 4.3 for an edge connected to an anchor node.

The first option makes use of the fact that we previously computed \mathbf{B} , which represents the original graph and all second order neighbors of each node. Consequently, all nodes sharing an edge with an annotation are also directly connected. If we would remove the annotation nodes from the graph, all nodes

of the same annotation would still be interconnected. Thus, the edges with an annotation node can be skipped while computing Eq. 4.3 but representations with the same annotation are still moved closer together, which resembles the work by [KTW⁺20]. We will refer to this option as *link* in the following evaluation.

The second option is to represent the annotations as vectors $\mathbf{a} \in \mathbb{R}^d$ in the embedding space. Then, we could treat the annotations as data when computing the loss function. However, the annotation vectors must be carefully chosen to avoid ambiguities. Here, we make use of the fact that any two random high-dimensional vectors are almost orthogonal to each other with high probability [BHK20], meaning there is almost no correlation between these vectors. This statistical behavior can be understood such that every annotation vector will have almost the same distance to any other annotation vector. When we compute the loss using these vectors, the data points belonging to different classes are trained to move away from each other. To implement these annotations, we introduce a high-dimensional L2-normalized Gaussian random vector $\mathbf{a}^{(c)} \in \mathbb{R}^d$ for class c into the batch which acts as the target z_t during computation of Eq. 4.3. The vectors are fixed at the beginning of the training. We will refer to this option as *anchor* in the following evaluation.

4.2.4 Experimental evaluation

We perform evaluation on two tasks. The first task uses data from the Illgraben deployments to detect rockfalls, earthquakes and noise. The second task is already known from the previous chapters and deals with detecting mountaineers from the seismic signal.

Illgraben Deployment

We evaluate the proposed approach on the Illgraben dataset by performing an ablation study and comparing the system to a random forest classifier, which is currently best practice for slope failure detection [WHv⁺21].

For the ablation study we use a classifier based on a single-channel variant of ResNet18 [HZRS15] as encoder in combination with an multilayer perceptron with 1 hidden layer as encoder head. During classification the encoder head is replaced with a classification head differing only in output size. Input to the ResNet18 is a log-compressed spectrogram of the seismic data. For more implementation details please refer to the Section 4.2.5. The ResNet18 model is trained with three methods, cross-entropy loss between the output of the classification head and the ground truth (Resnet18+XE), contrastive pretraining using the information graph (IG) and either the link (Resnet18+IG+link) or anchor (Resnet18+IG+anchor) strategy. Subsequently, the classification head is trained using cross-entropy loss. We compare training with system context information (SC) and training without it.

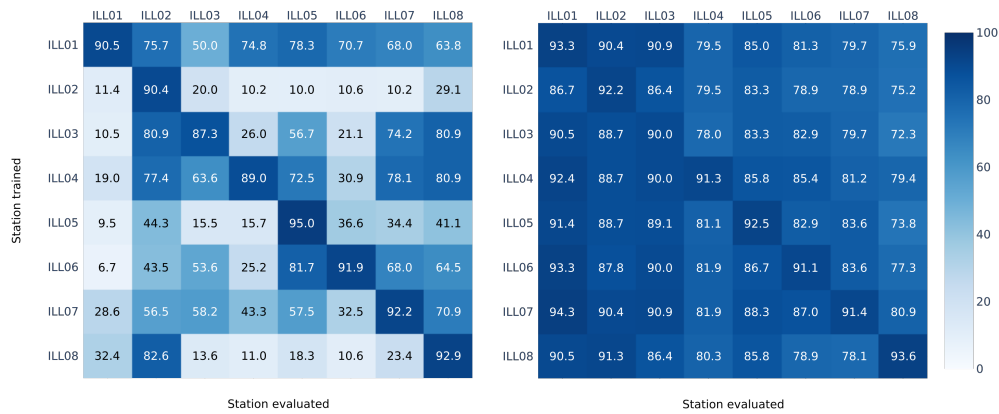


Figure 4.6 Each row indicates of which station the annotated training subset was used. Each column indicates the respective score on the subset of the test dataset for each station. (Left): Results for ResNet18+XE. (Right): Results for ResNet18+IG+anchor.

The benefit of our approach for the weakly-labeled setting is evaluated by using the available training annotations of all stations. The experiments are repeated 5 times and mean and standard deviation are reported. Robustness is evaluated by training the model variants when only a subset of the annotations are available. While the whole training data is available to train a classifier only the annotations for one of each of the 8 seismic stations can be used. All reported accuracies are based on evaluation on the test set using all stations and are reported as mean and standard deviation of all one-station evaluations.

The results presented in Table 4.3 show that the ResNet18 classifiers outperform the random forest classifier in the weakly-labeled settings (*all* and *all+SC*). The *all* column, illustrates that training using contrastive pretraining improves the performance significantly in comparison to the random forest classifier but only slightly in comparison to using cross-entropy loss (ResNet18+XE). However, if we include the system context information the accuracy improves significantly (*all+SC* column), demonstrating the benefit of additional sources of information for training.

In the robustness experiments (*one-station(+SC)*), all classifiers show a comparable bad performance of around 50% on average if only annotated data of one station is used (*one-station*) and the evaluation is performed on the test data of all stations. There is a significant variability in the results which can be explained with Figure 4.6 (left) illustrating that a classifier almost only performs well for the station it was trained on. If more non-annotated data from other stations is available, our method takes advantage of the system context information (SC) stored in the information graph (*one-station+SC*) and the average accuracy rises to over 84%. The increase comes from a better generalization to other sensors, as illustrated in Figure 4.6 (right). If non-

	Accuracy			
	all	all+SC	one-station	one-station+SC
Random Forest	86.4 %	-	n.a.	-
ResNet18+XE	91.3 % \pm 0.5	-	50.0 % \pm 15.0	-
ResNet18+IG+links	92.3 % \pm 0.3	93.8 \pm 0.3	44.0 % \pm 15.6	84.1 % \pm 2.6
ResNet18+IG+anchors	92.0 % \pm 0.4	93.9 \pm 0.6	47.9 % \pm 16.7	85.0 % \pm 1.8

Table 4.3 Classifier accuracies for different sets of available training annotations. Either all annotations (*all*) or only annotations for one station are available (*one-station*). Additionally, the information graph (IG) is used with or without system context information (*SC*).

	Accuracy	F1-Score
XE+Annotations (Baseline)	88.5 % \pm 0.9	88.9 % \pm 1.0
IG+Annotations	92.1 % \pm 1.0	92.8 % \pm 0.9
IG+Seismic	93.8 % \pm 0.3	94.5 % \pm 0.3
IG+Seismic+Annotations	94.2 % \pm 0.2	94.7 % \pm 0.2

Table 4.4 Accuracies and F1-Score for a mountaineer classifier using the Matterhorn dataset. The baseline is trained without the information graph using annotations and cross-entropy loss (XE). The other rows are combinations of system context information inserted into the information graph (IG). *Annotations* are anchor-based annotations and *Seismic* connects seismic segments overlapping in time

annotated data from other stations and system context information is available, our method increases classifier performance on all stations, thus demonstrating an increased robustness.

Matterhorn deployment

For the Matterhorn deployment we conduct a similar experiment in which we compare the impact of using different system context information. In this experiment, we compare how the different components of the information graph affect classification performance in comparison to a baseline which does not use the information graph for training. The general experiment setup is similar to the one used for the Illgraben Deployment, but we use a different dataset and limit our experiment to training with information graph and *anchor* annotations. Table 4.4 highlights that using the information graph with any system context information is better than the baseline. In the direct comparison of baseline and the system trained with information graph and only annotations (*IG+Annotations*), our approach performs better although the same amount of information is used to train the system. This increase is most likely due to the fact that our usage of the information graph introduces second order neighbors and thereby directly connects similar events. These connections can be seen as connecting "different views" of the same event type which can be exploited by the contrastive loss. This advantage of the contrastive loss becomes more obvious when we connect seismic segments overlapping in time (*IG+Seismic*) which improves the performance significantly in comparison to the baseline. The last experiment *IG+Annotations+Seismic* improves the performance again, although only little improvement is being observed in comparison to only using *IG+Seismic*.

4.2.5 Implementation Details

4.2.5.1 Details to Contrastive Learning with Information Graph

The seismic signals are subdivided into segments using a window length T_w and a stride T_h . The data subset for pre-training uses $T_w = 30s$, $T_h = 30s$, the subset for fine-tuning and the test set use $T_w = 30s$, $T_h = 15s$. Each segment's annotation is determined using the Illgraben event catalogue, which is split into training and test set with a ratio of approx. 70/30. Linear detrend is applied to the seismic signal before it is transformed into a log-compressed spectrogram with window length of 2.56 s and stride of 0.08 s. No data augmentation is applied. As encoder we use a single-channel variant of ResNet18 [HZRS15], without the final linear layer. The output of the encoder is a 512-dimensional vector, which is then passed through the encoder head, consisting of a multilayer perceptron with a hidden layer of size 512, batch normalization and ReLU non-linearity. The encoder head's output size is $d = 128$ and L2 normalized. During fine-tuning, the same encoder head architecture (with random initialization) is

Table 4.5 Random forest parameters

Number of trees	400
Split quality measure	Gini criterion
Minimum number of samples required to be a leaf node	1
Minimum number of samples for an internal node to be split	2

used as a classification head with an output size equal to the number of classes N_c .

In the supervised training we train encoder and classification head jointly using cross-entropy loss. For semi-supervised training (*all+SC* and *one-station+SC*), we train the encoder and encoder head with contrastive loss, then for fine-tuning we replace the encoder head with a classification head and train the classification head with cross-entropy loss while keeping the encoder weights fixed. In every epoch we first train with contrastive loss, then fine-tune the classification head. For optimization we use SGD with momentum 0.9 and weight decay 10^{-4} and a batch size of 128. The temperature coefficient τ is set to 0.1. We use a cosine annealing scheduler. In our experiments the edge weights of the information graph are 1. We perform a hyperparameter search to determine the best learning rate for each experiment individually. The individual parameters can be found in the source code accompanying this dissertation.

To counter class-imbalance, each batch contains the same number of examples for each class. During semi-supervised training the non-annotated data outweighs the annotated data by a factor of approx. 4.5 in each batch. We select our model based on a validation set which is 20% of the training set, except for the *one-station+SC* experiment. Here, we select model from the last epoch, since model selection on the one-station subset would deteriorate the generalization effect.

4.2.5.2 Details to Random Forest Classifier

Following [PHM17] and [WHv⁺21] we computed a total of 55 signal characteristics in the time and frequency domain, e.g., information on the signal form and dominant frequencies. For a complete description of the chosen features see [WHv⁺21]. We performed a three-fold-cross-validation grid search to optimize classifier performance. The resulting parameters are presented in Table 4.5.

4.2.5.3 Details to Matterhorn Annotation Set

The Matterhorn annotation set has been created by randomly sampling four images per day depending on image availability. The image and corresponding

2 minute seismic segment are then labeled and added to the set. Additionally, the training set (2016) contains selected period of known event activity. The result are two annotation sets, one for images and one for seismic data. In this chapter, we refine the seismic annotation set by annotating events instead of whole segments. The refined annotation set is generated by producing a bounding box for each event contained in a given segment. The bounding box might be confined to a subpart of the original seismic segment (for example if a helicopter is only apparent in the first 20 seconds) or go beyond its boundaries (if a helicopter started to appear before or after the segments start or end time, respectively). Using this procedure we verify every annotation of the test set. Note that due to this relabeling effort the accuracies reported in this section are not comparable to Chapter 2 anymore.

4.3 Conclusion

In this chapter we have presented methods to enhance domain expertise with machine learning by using unsupervised (Section 4.1) and semi-supervised learning (Section 4.2). Moreover, we have enhanced machine learning with domain expertise by incorporating system context information in addition to annotations to the learning process (Section 4.2).

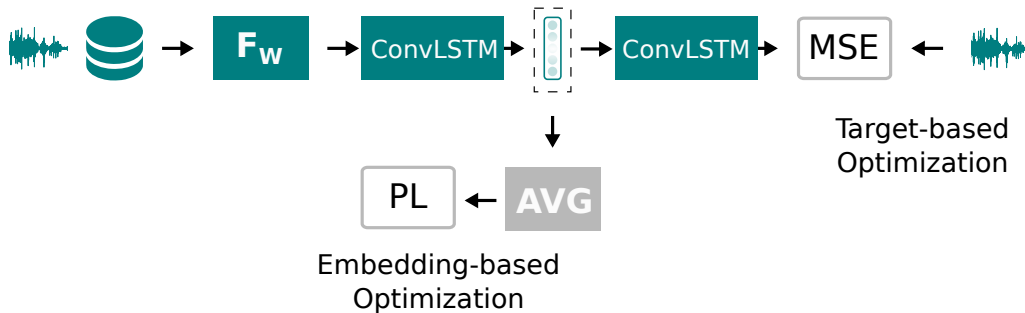


Figure 4.7 Data types, models and optimizations used in Section 4.1. Machine learning-based information extraction from acoustic data using data specific frontend (F_w), a convolutional long short-term memory neural network (ConvLSTM), a multi-layer perceptron (MLP) and average pooling (AVG). Target-based optimization is performed with Mean-Squared Error (MSE) Loss and embedding-based optimization with pairwise loss (PL).

We have demonstrated that in-batch similarity optimization can be beneficial to learn representations useful for classification and clustering. Figure 4.7 summarizes the information extraction system used in Section 4.1 in the context of this dissertation. We have presented a system including a Convolutional LSTM autoencoder to learn an embedding which is further optimized by an

embedding-based optimization method, namely pairwise loss, which amplifies inter-sample similarities. The embedding can then be used for tasks such as labeling, clustering or classification.

In Section 4.2, we have shown how similarity optimization can be improved by integrating annotations and system context information. As illustrated in Figure 4.8, the general structure of the information extraction setup is conceptually similar to Section 4.1 (Figure 4.7). However, there are significant changes in how the loss function is computed and that an information graph is used to embed system context information provided by domain experts.

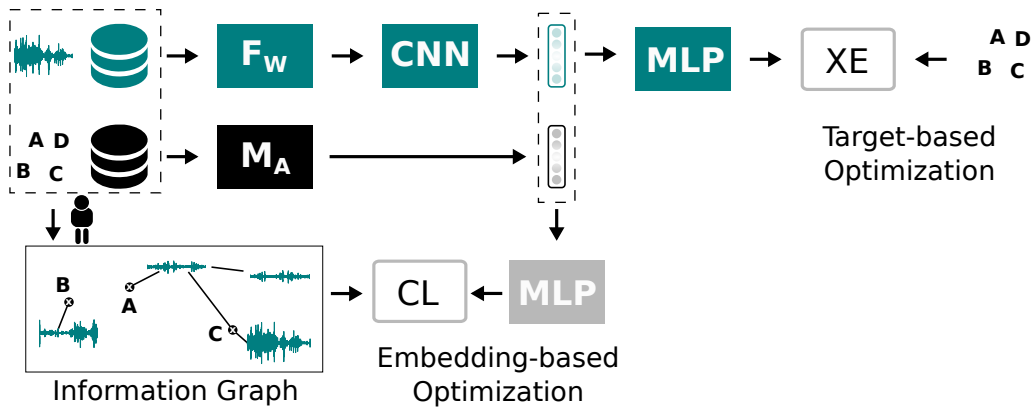


Figure 4.8 Data types, models and optimizations used in Section 4.2. Machine learning-based information extraction from seismic data and annotations using data specific frontend (F_W), a mapping from annotations to Gaussian random vectors (M_A), a convolutional neural network (CNN) and multi-layer perceptron (MLP). Target-based optimization is performed with cross-entropy loss (XE) and embedding-based optimization with contrastive loss (CL).

To conclude, we have presented a novel approach to learn with weakly labeled data for the case of mass movement monitoring. By using contrastive learning we can increase the classification accuracy compared to the reference implementation. Moreover, the presented method unifies data and annotation representations and thus inherently allows to integrate additional system information into the learning process. This additional information leads to a strong performance increase in a setting with limited annotations and diverging sensor characteristics, demonstrating increased robustness across sensors.

4.A Appendix

Related Publications

Unsupervised Feature Learning for Audio Analysis

Matthias Meyer, Jan Beutel and Lothar Thiele

Presented at the 5th International Conference on Learning Representations (ICLR) 2017, Workshop Track, Toulon, France

Contributions: Matthias Meyer developed the concept and discussed it with Jan Beutel and Lothar Thiele. Matthias Meyer developed the code, prepared and performed the experiments and evaluated the results. Matthias Meyer prepared the manuscript with contributions from all co-authors.



Paper

[arXiv:1712.03835v1](https://arxiv.org/abs/1712.03835v1) [cs.CV]

Using system context information to complement weakly labeled data

Matthias Meyer, Michaela Wenner, Clément Hibert, Fabian Walter and Lothar Thiele

Published as a workshop paper at the Workshop on Weakly Supervised Learning / ICLR 2021, online, 2021

Contributions: Matthias Meyer developed the concept and discussed it with Lothar Thiele. Matthias Meyer developed the code for the information graph with contrastive learning. Michaela Wenner developed the random forest classifier with contributions from Clément Hibert. Michaela Wenner and Fabian Walter collected and annotated the data. Matthias Meyer and Michaela Wenner performed the experiments. Matthias Meyer prepared the manuscript as well as the visualizations with contributions from all co-authors.



Paper

[arXiv:2107.10236v1](https://arxiv.org/abs/2107.10236v1) [cs.LG]



Project Webpage

matthiasmeyer.xyz/system-context-info/

5

foReal!

Team up for real-world data exploration

Previously, in Chapter 2, a methodology to annotate environmental data was presented. The main gist was that access to geoscientific expert knowledge is required to obtain reliable results but a certain amount of work can be accomplished by non-experts. However, without proper tools it is hard to comprehend environmental data for both experts and non-experts which negatively impacts analysis and development of models. In this chapter, we present a framework which is designed to bring together experts of different domains as well as the public to analyse and annotate environmental data by offering a comprehensive, cloud-based analysis platform. The platform can be used for data access, processing, visualization and annotation. Additionally, the framework is conceptually designed to allow the integration of edge computing making it a good candidate for the scenario presented in Chapter 3. Moreover, the platform is designed to blend in with the methods presented in Chapter 4, namely methods to perform model optimization with a limited set of annotations. A main contribution of Chapter 4 is the information graph, which allows to optimize a model by using a pair of data samples instead of requiring exact annotations. The platform presented in this chapter, features the flexibility to compare data pairs across a multitude of data scales and types and thus allows to find such data pairs.

Working with data from a real-world deployment requires a transparent and controllable data flow to avoid data cascades [SKH⁺21]. Data cascades are negative effects of faulty data processing during data analysis. Some examples are noisy data/annotations [SPI08, ZSS11], sensor misconfigurations and miscalibration [NSQ⁺18] or preprocessing artifacts. These effects may accumulate and distort the effectiveness of a trained model in a real-world context. Finding and solving all of these errors requires a diverse skillset, typically not found in small teams performing environmental research. One

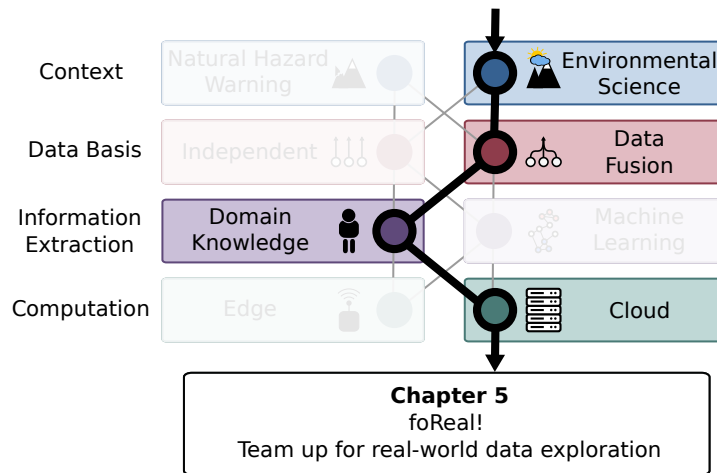


Figure 5.1 This chapter's position in the context of the dissertation

solution was provided in Chapter 2 which described a method for increased robustness against errors in the knowledge generation phase by systematically finding negative influences and integrating a human in the loop. Chapter 4 provided a solution for robustness against a shift in the data distribution by extending the training set with unlabelled samples.

In this chapter, we additionally focus on robustness against data corruption, missing data and misconfigurations during data processing. We also focus on avoiding misinterpretations during experiment design and analysis. By minimizing the effect of data cascades, we intent to improve reproducibility and facilitate semi-supervised learning with multi-type data. Through lessons learned from working with real-world data, a data analysis and visualization framework as well as a learning framework for semi-supervised learning were designed and continuously improved. We present the data analytics platform *foReal* which is be applied to the different types of geoscientific data obtained from a world unique scientific experiment in mountain permafrost conducted over more than a decade at 3500 m a.s.l. on the Matterhorn, CH [WBF⁺19]. The tooling developed allows to combine context data from different sensors and relate them to global models to anticipate the impact of climate change on third pole (high mountain) environments with a focus on enabling fast and easy exchange between application-domain experts and data scientists. Additionally, the framework supports the dissemination of information to the general public as one of many information channels for adaptation to climate change.

5.1 Introduction

A recent study [SKH⁺21] interviewed multiple data scientists on their problems during real-world data-analysis projects. The study concluded that, among others, problems *"occurred because of a default assumption that*

datasets were reliable and representative, and application-domain experts were mostly approached only when models were not working as intended [...]. With limited application-domain expertise, [data scientist] described how incomplete knowledge and false assumptions got incorporated into model building." [SKH⁺21].

As becomes apparent from this statement, analyzing domain-specific data, such as environmental data, demands a coordinated combination of domain expertise (application and data domain) and technology to define a model that can be automatically applied to long-term datasets. This expertise combination should be conceptually incorporated into the analysis from the very beginning to avoid mistakes and vain work. It should be expected that working with real-world data is error-prone such as data collected with a wireless sensor network [TKW20]. A fact which is signified in Figure 5.2 showing data availability (and non-availability) of a real-world deployment in a harsh environment [WBF⁺19]. In Chapter 2, we learned that compiling an annotated dataset for machine learning requires several iterations of labeling and label cleaning. While many annotation frameworks are readily available [Ama, Lab], not many frameworks allow for annotating multi-modal data [TMS⁺20].

To the best of our knowledge no frameworks exist for the set of data types used throughout this dissertation. This fact makes compiling a clean, comprehensive and representative dataset for long-terms, heterogenous monitoring deployment especially challenging and time consuming. Moreover, human-verification of a multi-modal dataset requires synchronized data sources [KBST12], allowing to analyze the data of different data types on different scales and in different representations. It is important that a visualization can be used to introduce the human/expert into the loop via labeling and reasoning [SKH⁺21]. Additionally, by making the data accessible and tangible to the general public as well as domain experts, a broader audience can support the knowledge generation (traditional or local knowledge [Hun00, CF13], identifying landmarks [VDOM19], providing eye-witness reports [DED⁺16, WJWH⁺18], providing domain knowledge about geophysical phenomena, contributing data). Finally, the gathered information can be used to train machine learning methods to automate high-resolution, short-term analysis for the whole long-term dataset and thereby contributing to a qualitative explanation of long-term geophysical phenomena. Therefore, large amounts of data need to be processed requiring suitable processing resources such as multi-machine compute clusters. The design of the tooling should allow to minimize the impact of data cascades during model development by giving the experts of complementing expertise access to all aspects of data processing and data analysis by offering clear interfaces and a transparent data flow. To avoid misconceptions it is important that experts have access to the same data representations as used for model development, which requires consistency between data for processing and visualization.

Last but not least, scientific work is required to be reproducible which should be

accounted for throughout the whole analysis process. Especially in the academic context, the first researcher analyzing a dataset is likely not the last. Therefore, a dataset should be regarded as an open, adaptive and evolving scientific artifact.

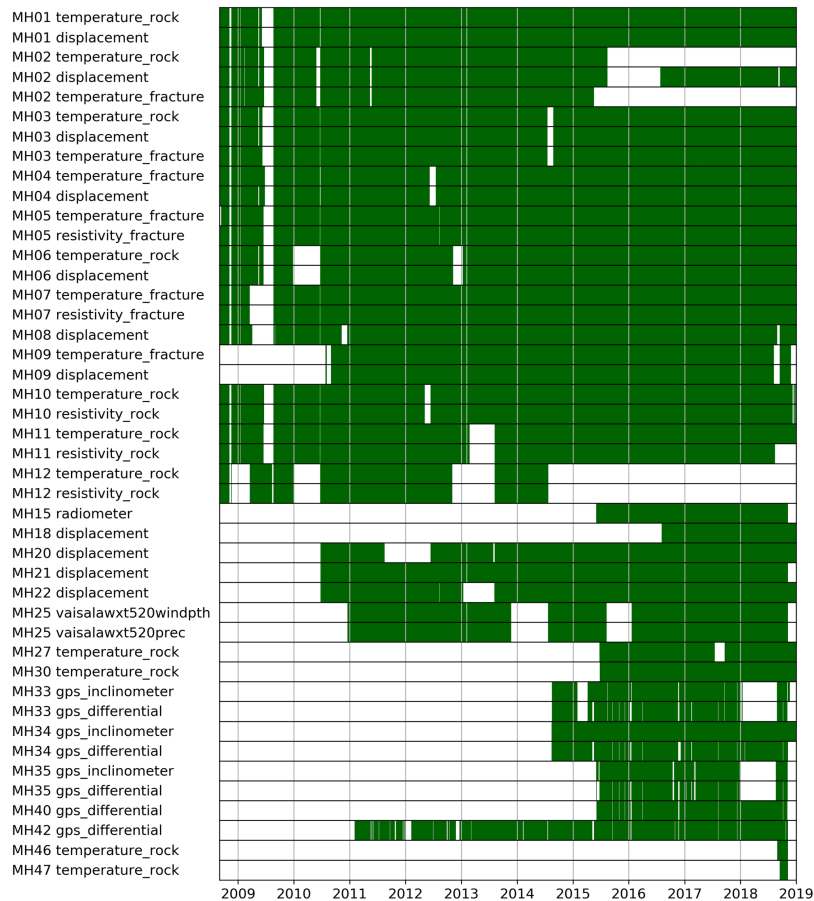


Figure 5.2 Data availability for the PermaSense Matterhorn deployment. The available data (green) is disrupted by significant gaps (white) demonstrating the error-prone nature of running wireless sensor networks in harsh environments. This image is taken from [WBF⁺19].

In this chapter, the main contribution is the data analysis platform *foReal* which is designed to satisfy the following requirements:

- Joint analysis of large, heterogenous datasets including weather data, timelapse photos, seismic streams and more
- Integrated human interface and feedback options such as visualizations or annotations
- Fast and efficient processing of large data amounts for human inspection or batch processing, scaling from single machine to multi-machine clusters
- Consistency between data for processing and data for manual inspection
- Separation of concerns: clear interfaces and transparent data flow

- Robust functionality when used with fragmented and erroneous datasets as well as continuously expanding datasets (stream of measurements)
- Consistency throughout analysis iterations (reproducible science)

5.2 System Overview

The use case for our framework is the analysis of data from a wireless sensor network. Before we can design a solution, we must understand the different components of such a system to define the requirements and challenges in terms of functionality and performances. As illustrated in Figure 5.3, on a high-level we distinguish (i) the data provider (ii) the data itself and (iii) the data user.

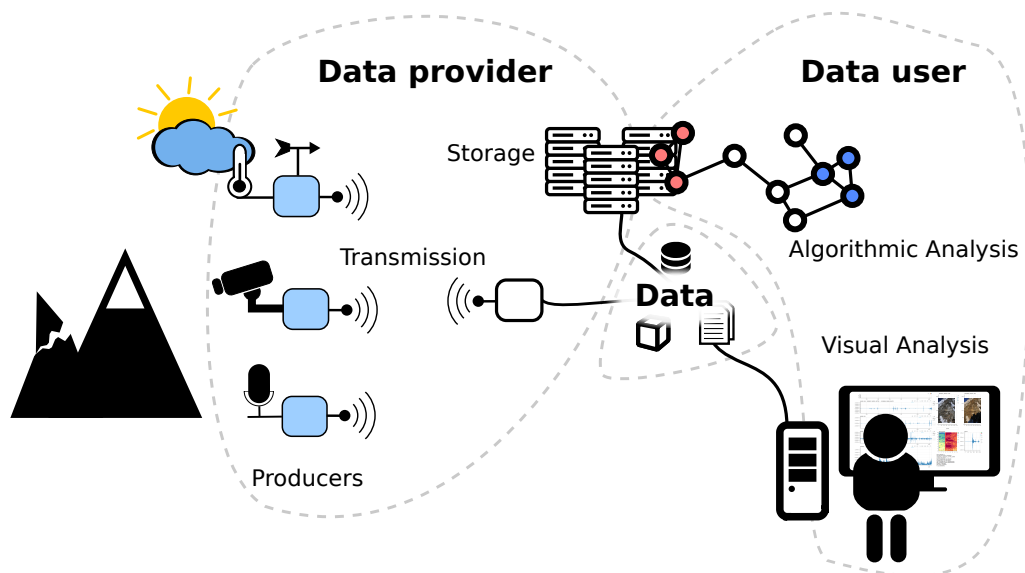


Figure 5.3 Components of a wireless sensor network deployment and analysis system

5.2.1 Data provider

On the technological side, the data provider is responsible for producing, transmitting and storing the data.

Producers The data is produced by sensors, which contain or are connected to a processing device digitizing the sensor signal and optionally applying signal filtering or other required processing steps. Certain settings of each sensor node are usually configurable, for example sampling type, sampling frequency, sensor calibration, detection thresholds. These can be configured pre-deployment or at runtime. Configuration changes

at run-time should be respected during analysis. Data integrity can also be affected by physical changes to the actual deployment such as replacing sensors, changing sensor locations, environmental changes, repeated short-term deployment campaigns (e.g. multi-year campaigns during summer season only). Additionally, data can come from sources not directly integrated into the wireless sensor network such as weather services, satellites or companies (see also Chapter 2). Additional data can also come from humans annotating the data, for example by providing eye-witness reports or labels.

Transmission The data transmission technology in a sensor network depends on bandwidth, distance and energy. Thus, the technology can vary between sensors depending on the use case, which has an impact on availability and latency. On the one hand, sensor data can be stored locally on a storage connected to the sensor and retrieved manually from the deployment. On the other hand, the data can be transmitted via wireless or wired technology. Typical wireless technologies include WiFi, Bluetooth/BLE, LoRA or custom, application-specific transmission protocols and modules. Connectivity issues are to be expected, due to the varying reliability of the wireless channel. Error-correction on the various layers of the transmission pipeline can prevent data loss, such as on-device buffering and backlog handling. However, these have implications on the latency and on the availability of data. On the server side of the wireless channel, the cable-based transmission between data storage and data user is a common bottleneck for large-scale data processing. In terms of latency and throughput, data on a remote storage leads to worse performance than data on a local disk. Additionally, fragmented files and non-aligned data (on disk and in memory) affect processing performance.

Storage The data produced must be stored in a suitable long-term storage which depends on the data amount, data shape and type as well as other factors like read/write speeds. The simplest way to store the data gathered with a sensor node, is to store data on-device and retrieve the whole device manually or the storage medium manually. This time-consuming process can be avoided by transferring the data produced by wireless sensor networks directly to a dedicated server. The choice of storage architecture depends on the data type. For example time series data such as system information, weather data and other table-based, structured information is usually stored in a *relational database* (SQL). If data is not table-based but unstructured, such as meta-information, a *NoSQL database* is a better choice. Other file types require again a different storage architecture, such as large binary files for which databases are an inefficient storage type. Alternatively, large binary data can be stored on a simple filesystem, it is however more efficient to store it in an *object storage* (Amazon S3, Azure Blob Storage). However, these storages can only read/write whole (large) files at once. When only a

subsets of a large file should be read/written, it is more performant to store them using a *chunk storage*, which subdivides any large binary file into a sequence of smaller blocks. These chunk storages are beneficial for an ever-growing dataset due to their inherent ability to scale-out. They feature a virtually unlimited object size, blocks can be stored across distributed machines and blocks can be stored using object storages. However, chunk storages come at higher I/O utilization cost.

5.2.2 Data

The data produced by a wireless sensor network tend to vary in terms of

- Data type (images, timelapse images, seismic, audio, time-series, ...)
- File format (jpg, png, miniseed, csv, zarr, ...)
- Data shape (tables, trees, graphs, cubes, text)
- Sampling type (asynchronous, continuous, event-driven)

These varieties results in an increased complexity and susceptibility to errors since individual methods for data loading, transformation, storage and analysis are required. Note that in this chapter the term data type does not define programming constructs such as *integer*, *string*, *float*.

We distinguish between a physical data model and a logical data model. The physical data model is the representation of data in the physical world, for example files stored as bits on a filesystem. The logical data model is a high-level abstraction of data for example tables, graphs, cubes. This logical data model is useful for working with data without the need to know their representation in the physical data model.

A dataset is a collection of data. The assignment of the term dataset to a persistent, never changing collection of files, as often used in machine learning research, may be good for algorithm development or benchmarking but does not reflect our scenario. In the upcoming sections, we will develop a different definition of a dataset in which we will define it as data sources, optionally transformed by processing steps (for example resized timelapse images, spectrogram of seismic data) in combination with a formalization of the dataset extend (for example which set of images, which time periods of the seismic stream).

5.2.3 Data user

The data user may be broadly categorized into visual analysis performed by human/experts and algorithmic analysis carried out by computers. These two subcategories have partially diverging requirements regarding data usage. Visual analysis needs to account for the fact that environmental effects manifest themselves in multiple scales and different data representations. In visual

analysis a continuous data segment is usually accessed at once but it may consist of several billion datapoints. Here, the bottlenecks are disk access, rendering time, memory size of rendering products and for web-based visualization tools also transmission capacity. In contrast, algorithmic analysis usually accesses a data in batches, which relieves the memory requirement but disk access, processing time and access to dedicated hardware (GPU) are major bottlenecks. Machine learning typically requires random access to the data during its training phase and sequential access when applying the trained model to the data. Since the same data is loaded repeatedly during training, loading excess data should be minimized (for example by loading low-resolution images instead of loading high-resolution images and then resizing them).

5.2.4 Motivational Example

In this chapter, we will explain the concepts and implementations of the *foReal* analysis framework using an example based on the setting of the previous chapters, namely mass movement detection with a wireless sensor network in high-alpine permafrost. Given is sensor data from the PermaSense Matterhorn deployment [WBF⁺19]. We would like to analyze mass movements using the spectrogram of a seismic signal while reducing the impact of rain, wind and mountaineers using expert knowledge and machine learning.

5.3 Concepts

5.3.1 Transparent separation of concerns

In environmental monitoring, a diverse set of tasks need to be performed such as experiment and system design, instrument development, deployment setup and maintenance, data collection and management, algorithm development, model definition and data analysis. These tasks require different expertise which can be found in different domains for example hardware domain (Develop, deploy and maintain wireless sensor network), software domain (efficient data storage, data transfer, data management), application domain (define application scenario, set up experiments, contribute expert knowledge, analyze data) and data domain (find adequate data analysis methods, automate data analysis). Some or all of these domains may be covered by one individual but usually a team is required to accomplish an experiment.

On the software side, data generation, processing and analysis can be subdivided into multiple components each requiring specific expertise. However, errors in each component may have a strong negative impact on the overall analysis and combatting these errors requires again a very specific set of expertise. Unfortunately, certain types of errors don't necessarily reveal themselves during development of each component but in later stages when all components are

combined.

To address the challenge of a multi-disciplinary data analysis project, we define the following software requirements:

Separation of concerns Abstraction of a system into **modules** which are each responsible for one specific subtask of the system. Each module can be worked on individually which allows to distribute the work to experts. Each module must receive and provide their data in a common, well-defined format. **Clear interfaces** are necessary to join modules to a functioning system.

Transparency Each user or developer should be able to trace errors back to their origin in all modules of the system.

Ideally, these requirements are reflected in the analysis workflow and software package. Figure 5.4 highlights a data flow graph for our motivational example. Each node of the graph represents either a processing unit (light-purple) or a human-lead step (dark-cyan). A data access unit for each respective sensor should be designed to give access to the data and include meta information (for example about data availability, data validity or sensor peculiarities). This unit acts as a bridge translating raw or processed sensor data to a common data shape used for analysis. Next, the data processing is split into well-defined processing units, thus distinguishing functionality and implementation. Each processing unit processes the data and returns it in the common data shape. In the example, seismic, image and weather unit as well as annotations can be used to train a model, which can subsequently be used to create predictions for long-term analysis. Annotation errors can be mitigated by using the visualization tool for relabeling the annotation set. Additionally, the visualization tool can be used to perform analysis.

The advantage of a data flow graph described in Figure 5.4 is that it can be implemented as a processing graph, similar to filter graphs in signal processing, where each node transforms the incoming signal(s) and delivers them to the next node(s) in the graph. Processing graphs are easy to understand and use even with little technical expertise. In our framework, graph computation is performed using `dask` [Dasb], a library for parallel computing. The `dask` framework allows to schedule the graph's tasks in a single thread or parallelize the graph on one or many computers. Thus, we can adopt `dask` capability of scaling out (using more machines) if our analysis task or dataset grows. `Dask` is comparable to Apache Spark [Apa]. Both implement computation using a directed acyclic graph [Dasa]. Spark is an extension of the MapReduce paradigm [DG08] and thus lacking flexibility to more complex algorithms. Moreover, while Spark is strong in business intelligence it is lacking good support for scalable multi-dimensional arrays which are crucial for our application. `Dask` is based on a generic task scheduling and supports more complex algorithms and as well as multi-dimensional arrays.

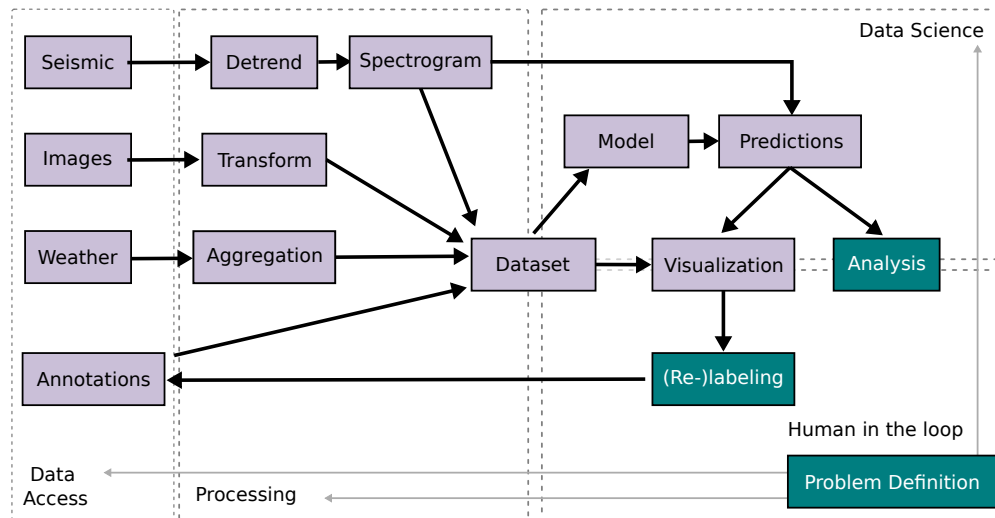


Figure 5.4 Data flow graph for our motivational example. Software components are highlighted in light-purple, human-lead steps in dark-cyan. Data access units bridge sensor data and processing units. By providing the output in a common data shape, multiple units can be concatenated. The output of such concatenations can be combined into a dataset, which can be used for visualization or for developing a model. Finally, the human in the loop can perform analysis or relabeling. Such data flow graphs provide an easy to understand high-level view of a software solution.

5.3.2 Declarative Data Access and Data Processing

To bridge expertise domains it is more relevant to define *what* is needed (declarative), instead of *how* it is implemented (imperative). In the case of our motivational example, *what* is needed for analysis is the time-frequency representation of seismic data for a given time period. Ideally, *how* the complex data processing leading to a spectrogram is implemented could be abstracted away and hidden from the data user. To illustrate the complexity, generating a spectrogram would include the development of a program to determine which and how much waveform data is needed to produce a spectrogram of requested size. Then, the software would need to check if the waveform data is available, loading waveform data from disk and transforming it with an efficient implementation of a short-time fourier transformation. It is important to consider that the spectrogram computations uses a sliding window, thus its output dimension is depending on the sliding window parameters (window size and stride). If the waveform duration is not chosen according to these parameters, the spectrogram will not have the desired dimensions.

Here, a combination of processing graph and declarative approach can be beneficial. First, the processing graph provides a generic framework for how a spectrogram is computed. Secondly, we could declare which time segment of a spectrogram is requested instead of declaring which time segment of waveform to transform into a spectrogram. A designated spectrogram unit would hide the

complexity of choosing the correct waveform duration for the given spectrogram parameterization to generate the spectrogram of requested size.

```

1  {
2    indexers: {
3      time: {
4        start: 2017-07-02T10:01:00,
5        stop: 2017-07-02T10:02:00},
6      frequency: {
7        start: 0,
8        stop: 20},
9      sensor: [MH36, MH38],
10     },
11    config: {
12      global: {
13        raw: False}
14     },
15     types: {
16       Seismic: {sampling_rate: 100}
17     },
18     keys: {
19       spectrogram_id: {stride: 1024}
20     },
21   }

```

Figure 5.5 Example of a request and its structure. Such a request can be defined by the data user or be the result of a processing unit’s configuration function.

L2 `indexers`: Define which data segment to return

L3 `indexers.time`: Select a range on the time dimension

L6 `indexers.frequency`: Select a range on the frequency dimension

L9 `indexers.sensor`: Select two specific values on the sensor dimension

L11 `config`: Define global, type-specific or unit-specific configuration

L13 `config.global`: Affects every unit with this parameter

L16 `config.types`: Affects all units of specified type (Seismic)

L19 `config.keys`: Affects the unit with the specified key (`spectrogram_id`)

In *foReal* a declarative procedure is implemented with so called *requests*, exemplified in Figure 5.5. A request is a metadata information which contains parameter information for selected units in the processing graph. These parameters can for example include, which time period to load, from which seismic station to load the data or which preprocessing parameters to use. Parameters declared in a request can be valid for all processing units in the graph (*global*), for a processing unit type (*types*) or one specific instance of a unit (*keys*). Moreover, it includes parameters which act on data, such as the *indexers* parameter, which contains information about which subset of the

requested data segment is relevant.

Figure 5.6 illustrates how a multi-modal data sample is computed using *foReal*. In the figure, a processing graph is given which contains a path for computation of a seismic spectrogram and one path aggregating the rain intensity of the last hour. Most of the parameters of each processing unit are predefined during graph design and are not explicitly mentioned in the given requests. The request #1 is designed to extract one time segment from the processing graph. It is propagated towards the *Seismic* and *Weather* data access units. Internally, the *Spectrogram* and *Rain Aggregation* unit update the request with a new start time and forward it. Additionally, the *Rain Aggregation* unit requests only the rain intensity subset, given that the *Weather* unit provides a set of different weather measurements (for example rain intensity, wind speed, ...).

More generally, configuring a processing graph using a request proceeds in two phases, a configuration phase and a computation phase. In the configuration phase, the respective request is propagated through the processing graph from sink to source. Each unit receives the request from its successor (in direction towards the sink). Simply put, for every unit the following question must be answered: *"What does this unit need from the previous units to fulfill the given request?"* This means each unit extracts and verifies from the request the metadata relevant for it. It then adds, modifies or creates requirements it needs from its predecessors. Each unit can also indicate if it requires the updated request in the computation phase again. The updated request is then forwarded to the next unit until all sources of the graph are reached. If nothing is requested from previous units, all previous units (if any) can be removed from the processing graph. In the second phase, the graph is scheduled for computation, starting from the data access units to the sink units. Each unit receives multiple inputs, including data from the previous unit(s) and the request from the configuration phase.

As explained before, processing the data can be a complex, error-prone task, especially if the data source is sensitive to errors. The declarative approach can potentially reduce these errors because it provides a clear interface to gather the relevant information. Using a declarative approach allows to disconnect conceptual errors (Is a spectrogram the right choice to analyse my data?) from implementation errors (Is enough waveform data loaded to get the spectrogram of required size?)

We can now redefine a dataset in the context of this chapter:

Dataset A processing graph combined with a request or set of requests.

A dataset explicitly includes not only data but also processing steps increasing its flexibility to address errors and facilitating reproducibility.

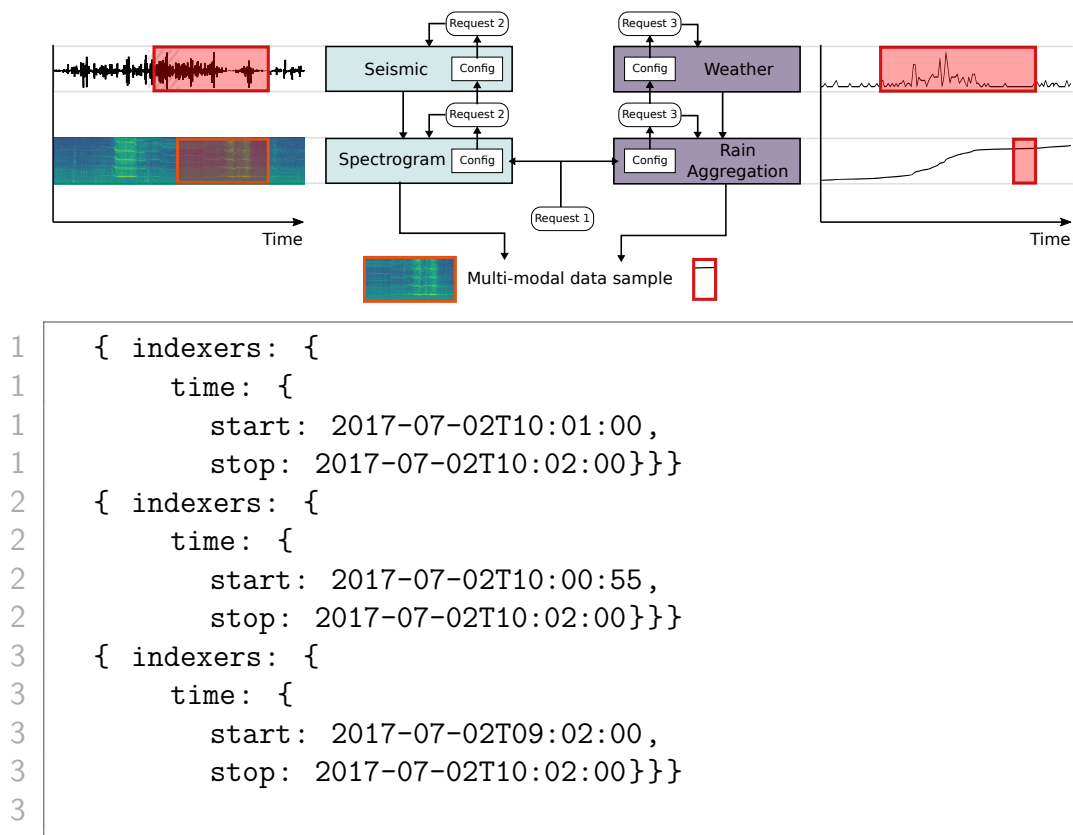


Figure 5.6 Exemplary illustration of the configuration procedure (top) and the corresponding requests (bottom). Each request number in the graph (#1,#2,#3) corresponds to the respective line number in the textbox. A processing graph containing a path for seismic spectrogram computation and one path for rain aggregation is given. A request (#1) is propagated from the graph’s outputs towards the data access units *Seismic* and *Weather*. Each processing unit configures itself and updates the request before forwarding the request to the next unit. In this example the units update the request’s start time according to their internal requirements (request #2 and #3). Additionally, the *Rain Aggregation* unit requests the *rain_intensity* subset of the weather sensor in its request (#3). The units *Seismic* and *Weather* just propagate the requests (#2 and #3, respectively) through without modification to receive them as input. Only request #1 is created by a data user, the other two requests are abstractions created by the respective processing units.

5.3.3 Data Independence

The declarative approach abstracts the physical data acquisition (vibration registered by a sensor and digitized) into a logical data access pattern (request signal from time t_0 to time t_1). Similarly, we want to abstract the physical data model (for example a file stored in a filesystem on hard disk) into an easy to understand logical data model (for example a table). This abstraction is usually referred to as data independence [BNK16].

Focusing on data models is crucial in long-term monitoring projects since

data from such projects is ever evolving. Over the course of a long-term measurement, changes in file formats or storage engines are likely. Moreover, changes in data types can occur when for example new sensors are installed.

The *foReal* framework is designed to work with specific logical data models for data and annotations which will be described in the following.

5.3.3.1 Data Cubes

Environmental data is inherently multi-dimensional. Examples include

- time-series (1-dimension: time)
- seismic spectrograms (2-dimensions: time,frequency)
- images (3-dimensions: x, y, color)
- timelapse images (4-dimensions: time,x,y,color)
- seismic spectrograms in global context
(7-dimensions: latitude,longitude,elevation,station,channel,time,frequency)

The logical data model for these kind of data are typically multi-dimensional arrays or so-called data cubes. In environmental monitoring, often the data along a dimension is meaningless without additional information, such as timestamps for the time dimension, frequency steps for the frequency dimension, or coordinates for spatial data. Therefore, data cubes for environmental data require the option of additional one dimensional arrays (from here on called coordinates) describing the content along each dimensions. These so-called labeled multi-dimensional arrays are implemented efficiently in the xarray framework [HH17], which is used internally by *foReal*.

foReal's request-driven data access is grounded in data cubes. A request in *foReal* slices out a segment of a multi-dimensional data cube using coordinates to select the requested data segment.

5.3.3.2 Annotations

We want to achieve data independence for annotations, meaning instead of defining annotations by their representation in the physical data storage (name of the folder containing images of respective label) we want to abstract it into a logical data model (directive to connect data with a label). This requirement comes from the fact, that the we need to be able to restructure or resample the data without losing the annotation information. For examples if the annotations are stored as folder names (as done for certain image datasets [DDS⁺09]) they are attached to the storage structure (filesystem). When we change the storage system or just restructure the data by sorting it differently the annotations information might be lost. Moreover, annotation should be reusable and independent of data type. If we, for example, annotate

wind in a time-series stream, we should in principle be able to apply this annotation to a co-located seismic sensor.

In *foReal*, annotation can be connected to the respective data by using a similar approach as used for *requests* in the previous section. Like a request, annotation contain an *indexers* key defining the concrete slice of data to be annotated. In addition, it contains a *targets* key containing the annotation. We do not put harsh constraints on what a target is but usually it would be the class name of the annotated event. The annotation might additionally contain more information such as how or by whom the annotation was created or an event id to trace and track multiple annotations of the same event, for example to annotate the same mountaineer on multiple, consecutive timelapse images. The annotation format is exemplified in Figure 5.7 using images and seismic data. The figure shows graphically which segment of the image or seismic stream is annotated. In addition, three annotations are given in JSON format.

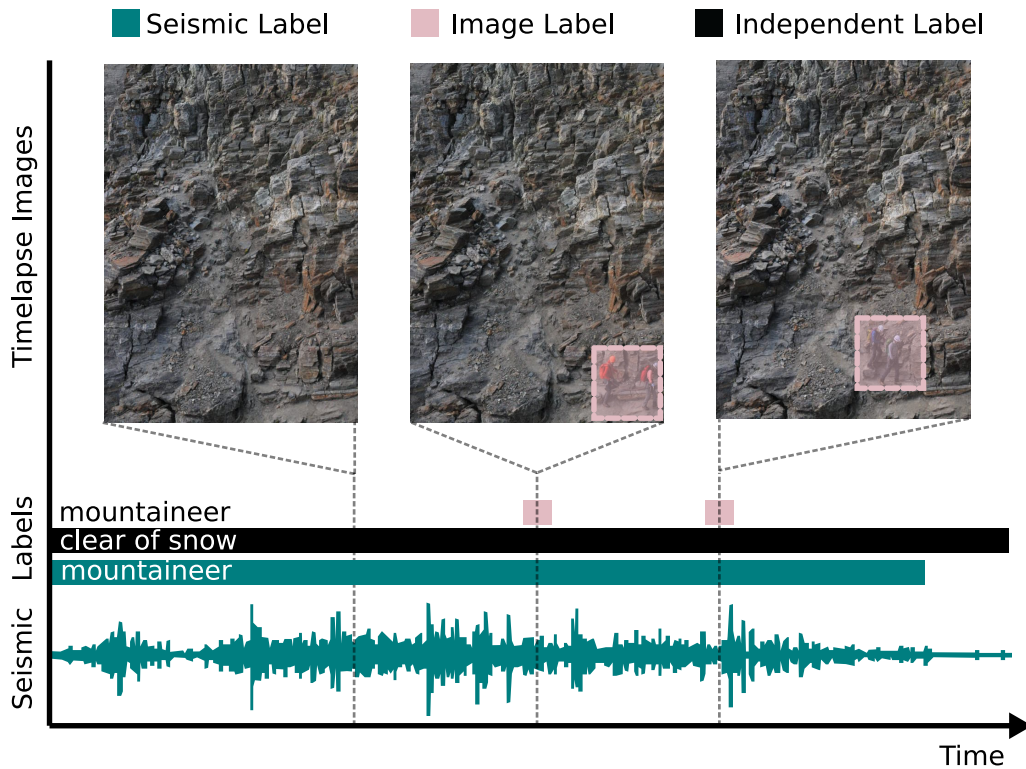
foReal annotations can be used for two purposes. First, a given annotation can be used to load the corresponding data segment. Second, a given data segment can be annotated with the corresponding annotation(s).

Figure 5.8 a) depicts how an annotation can be used to load a data segment. Since each annotation contains the same information as a request, it can be used as-is to request the annotated data segment. Moreover, it can be used to retrieve correlated sensor values. For example, we can use the seismic annotation given in Figure 5.7 to request all correlated images by replacing the annotation's *sensor* field with the sensor name of the camera.

Annotating a given data segment with annotations is a common scenario, for example when multiple mountaineers are annotated on an image and we want to crop the image while retaining the annotations within the cropped area and discard the annotations which are outside. This scenario is more complex because we need to be able to find the annotations describing a data segment of arbitrary size.

In our framework, the data segment can be defined by a request. A request can be regarded as bounding box. For the case of a one-dimensional time series, the "bounding box" would consist of start and end time. For the case of a cropped image, the bounding box would consist of the cropping area (start and end values on both, horizontal and vertical axis). In general, a request can be regarded as a multi-dimensional axis-aligned bounding box, which means it consist of a start and end value for each dimension of the array. Similarly, an annotation can be seen as such a bounding box. It describes what (event, object, ...) is inside the bounding box area.

The task is to find all annotations of the annotation set which are within the request's boundaries, meaning we need to find all annotation bounding boxes that overlap with the request bounding box (illustrated in the top path of Figure 5.8b)). We implement the multi-dimensional overlap detection iteratively by first checking for overlap on the first dimension between request



```

1 { indexers: {
1   time: {
1     start: 2016-08-04T11:44:17,
1     stop: 2016-08-04T12:09:32},
1     sensor: [MH36, MH38]},
1   targets: {mountaineer: true}}
...
4 { indexers: {
4   time: [2016-08-12:04:12],
4   x: {
4     start: 95,
4     stop: 145},
4   y:{
4     start: 20,
4     stop: 70},
4     sensor: [MHDSLRL]},
4   targets: {mountaineer: true}}
5 { indexers: {
5   time: {
5     start: 2016-05-02,
5     stop: 2016-09-15},
5   },
5   targets: {
5     clear of snow: {annotator:mountain-lodge keeper}}}
...

```

Figure 5.7 Sensor-specific label or sensor-independent annotation can be described using the same syntax. *indexers* describes the actual data slice to be annotated. *target* contains the annotation.

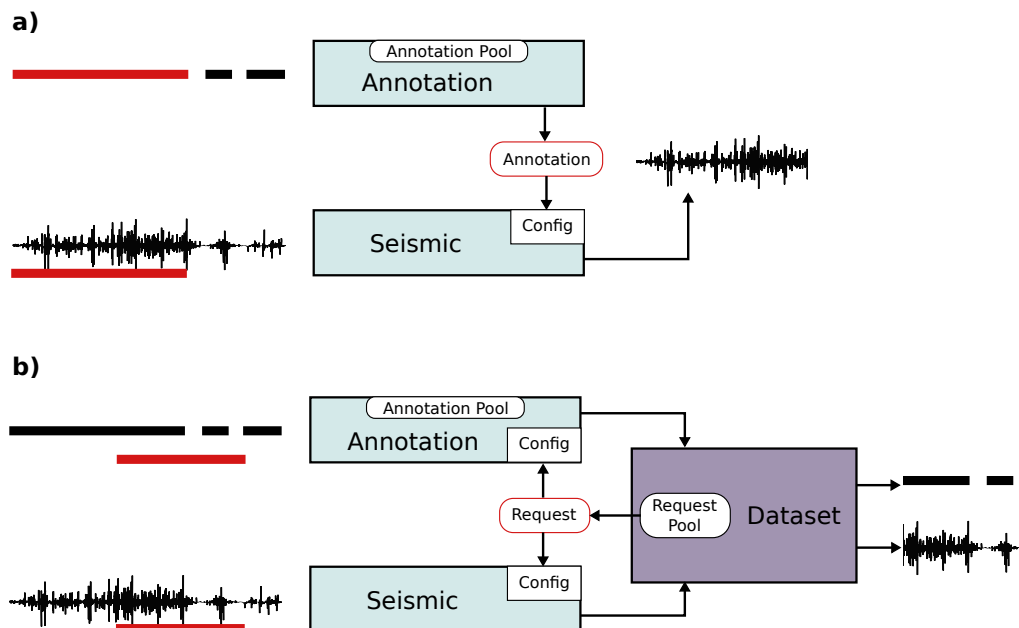


Figure 5.8 Usage of annotations in the context of *foReal*. The request's start and end time are highlighted as the red line below the signals. In **a)** an annotation is used as a request to load the corresponding seismic segment. In **b)** a request from a dataset is used to load a seismic segment and all corresponding annotations. The upper path in textbfb) demonstrates how a request (one red line), which can be regarded as a one-dimensional "bounding box", overlaps with the annotation's one-dimensional "bounding boxes" (three black lines).

and annotation set. The other dimensions will only be checked for the pairs with an overlap in the first dimension. Conceptually, the way a request slices out an annotation segment out of an annotation set is similar to the way a request slices a data segment out of the data cube as illustrated at the bottom of in Figure 5.8b).

5.3.4 Processing Failures

As highlighted before, the data analysis will most probably contain errors. We therefore incorporate this fact into the software design process and provide safeguards against processing failures at any stage of the processing graph. Corruptions in specific data segments should not affect the processing of other segments. In *foReal* a failsafe mode is implemented which catches all errors without disrupting the overall data process while providing options to trace back the error to its source. It provides options to ignore errors, raise a warning or fail upon error. Additionally, in a distributed system workers are likely to fail. The dask framework being used by *foReal* allows to mitigate worker failures while performing all computations correctly.

5.3.5 Potential Use Cases

There are several advantages using a declarative data access model as presented in this chapter, which are listed below.

Flexibility Selecting the required data segments from a dataset is facilitated, since complex data loading and processing is abstracted.

Inter-dependencies If there is a dependency of a successive task on its predecessors the declarative paradigm gives the successive task the option to declare these dependencies. Examples are the dependency of spectrogram input to its parameters.

Data augmentation Declarative data access can be used for controlled randomization. Each parameter can be defined and stored on a per-sample basis, which allows perfect replay and per-sample optimization of data augmentation parameters.

Dynamic sample merging Deep learning for audio applications benefits from superposition of multiple audio segments during training. For example merging two segments of the same label create a new, unseen segment while retaining the correct label. Declarative data access facilitates the process because a second segment of the same label can be explicitly requested during runtime.

Dynamic zoom Signal processing algorithms usually scale with input size, meaning if we process an image, the runtime is dependant on the image dimensions. In Chapter 2 we avoid large processing costs by downscaling the images before applying an image classifier. However, small features are indistinguishable in low-resolution images. Zooming in on details could reveal more information but only if zooming results in a higher resolution segment of the image. With declarative data access we could make the resolution dependent on the zoom level - keeping the image dimension constant while the resolution scales.

Access to expanding datasets Continuous data acquisition requires certain computations to be run every time new data arrives. Declarative data access avoids these time-triggered computations because when the data is accessed the computations will be run automatically.

Relabeling During relabeling new annotations are added to the dataset or the existing annotations of a dataset are modified. If the declarative approach (for example requesting all annotations of a time segment) is used in combination with the data independence paradigm (annotations are abstract bounding boxes), the change in the annotation set is immediately reflected in any use case of the dataset (machine learning, visualization), which makes the different use cases consistent.

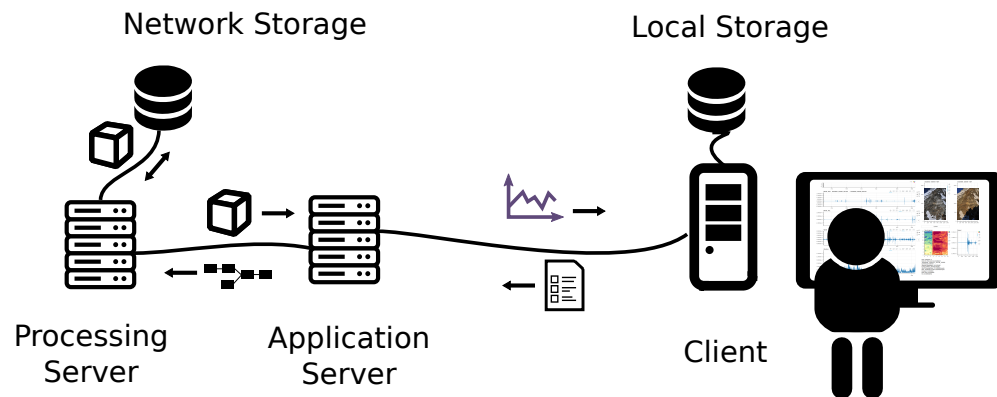


Figure 5.9 Illustration of the *foReal* visualization web service and general computing service. The visualization service runs as a web app on a dedicated application server and can be accessed via the client's browser. If the client selects a data segment to be plotted, a request is generated and the respective computation graph is configured on the application server. The configured graph is then sent as a task to the processing server, which decides what needs to be computed and what can be loaded from disk. The computed/loaded data is then sent back to the application server which subsequently generates plots to be served to the client. The application and processing server can also fulfil processing requests independent of the visualization service, for example for machine learning applications.

The main disadvantages is the computational overhead due to the configuration phase before each computation. This overhead becomes significant for large graphs since the configuration needs to traverse the whole graph.

5.4 Implementation

This section describes the major components of the *foReal* framework. We focus mainly on the components which are important for the Matterhorn deployment. Here, data access units are the primary bridge to the deployment's sensors. These units are often required to fetch data from remote storages, for example to obtain PermaSense data [Per] or accessing the Arclink service [Arc] for seismic data. Examples of data access units implemented in *foReal* are *Seismic* which loads seismic data from disk or webservice, *Image* which loads timelapse images from disk, *CSV* loads and converts an arbitrary comma-separated-values file, *Annotations* which loads annotations from disk and performs overlap detection.

Figure 5.9 highlights the computational setup in which a Processing Server, Application Server, Client, Network Storage and Local Storage are used for visualization and data processing. Note that processing server, the application server and the storage can also be used without a dedicated user interface, for example to for machine learning applications.

As indicated in Figure 5.9, processing and application are conceptually separated. In practice, the processing server is a dedicated process running on a dedicated machine, a cluster or even on the same machine as the application. Distributed processing as well as task scheduling is performed using the dask framework. The dask scheduler can dynamically respond to requests from multiple clients, for example scheduling a machine learning training run while computing data for distinct plots of multiple concurrent visualization sessions. Since we are dealing with larger-than-memory datasets, the main performance bottlenecks in the system of Figure 5.9 are available resources (system memory, disk space, GPUs) and data transfer (from/to disk or via network). Especially the server to network storage link affects the processing performance. Moreover, the application server to client link affects the responsiveness of the visualization tool. In *foReal*, several steps are taken to address these bottlenecks, such as caching of intermediate results or precomputing plots to minimize the data transfer between server and client.

5.4.1 Persist

Unlike a cache which stores the data temporarily in memory, the *Persister* unit stores the processed data permanently on disk in an efficient manner. The main use cases for using a *Persister* are (i) if the result of a costly computation is reused repeatedly but does not fit into memory, (ii) if access to the original data is slow (for example if it resides on a remote storage) and faster storage is available and (iii) if the data is only available in an inefficient format (for example in a large CSV-file).

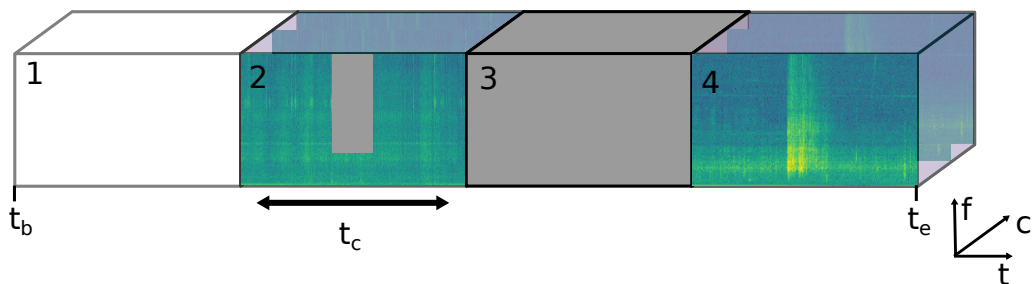


Figure 5.10 Illustration of how chunks are stored with a *Persister*. The persister is initialized with a dynamic dimension (*time*), dataset boundaries (t_b to t_e) and a chunk size (t_c). The sizes of the other dimensions, (here, *frequency* and *channel*), are automatically inferred from the preceding unit in the processing graph (Spectrogram). Chunks are empty until accessed at least once. A chunk can be "not loaded yet" (1), fully loaded but partially corrupted (2), fully corrupted or no data available (3) or correctly loaded (4). Only chunks 2 and 4 consume disk space.

In *foReal* the *Persister* is implemented as a unit in the processing graph, independent of any data type. It can be placed at an arbitrary position in the processing chain. Therefore it is possible to persist data access units as well

as intermediate processing representations (for example a spectrogram). The persisters storage pattern is illustrated in Figure 5.10. Since `persist` works with chunked data cubes it is independent of the actual data type. The *Persister* is designed to build up the storage incrementally, therefore disk space is only allocated for data segments which have been requested at least once. This incremental build up can only be performed along one, predefined dimension, the so-called dynamic dimension. All other dimensions of the multi-dimensional data must be fix in shape and coordinates. The *Persister* is initialized using a data prototype, which is obtained by calling the *Persister*'s predecessors in the processing graph with a prototype request. Based on the data prototype the data cube structure, including shape, coordinates and chunk sizes is defined. The meta-information about the structure is written to disk but no data is written to disk yet. When a data segment is request from the *Persister*, the *Persister* seeks the related chunk and if it is not available requests the required data from its predecessors and the chunk is written to disk. If the chunk exists it is forwarded to the next unit. The data is stored in the zarr [Zar21] file format, which is an efficient format for multi-dimensional data. On top of using an efficient file format, we make use of caching to store recently used chunks in memory which may improve performance.

The advantage of the *Persister* with its incremental data loading is that we often do not want to preprocess the whole dataset if we are only looking at a fragment of the dataset. For example we mainly work with data from one year and might sporadically need one hour of data from another year. With the *Persister* we can balance the storage to processing time trade-off. We can store the items we need frequently for fast access but we do not need to store everything while still being able to dynamically load from other time periods using a consistent interface.

To highlight the benefit of the *Persister* we have conducted benchmarks resembling a machine learning training scenario. For the benchmark one hour is subdivided into 120 segments of 30 second duration. For each segment the seismic data of one of three channels is loaded, adding up to a total of 360 data samples being loaded. In one iteration all 360 data samples are randomly loaded. All benchmarks are run for two iterations.

The benchmark is performed on a machine with CPU AMD Opteron 23xx running at 2GHz (16 core) and 16GB RAM. The data is loaded and stored on a local solid-state drive (SSD). The seismic data is stored in miniseed files (one file per hour per channel) on the same disk as the persisted data. The persister's chunks are aligned with the miniseed files, meaning the time period stored in a file is the same as stored in a miniseed file. The data is chosen such that two chunks must be accessed to load one iteration. We conduct the following 5 benchmarks:

1. **Waveform**: The seismic segments are loaded from the miniseed files via a *Seismic* data access unit.
2. **Waveform+Persist**: The *Persister* module is attached to the *Seismic* data access unit and the segments are loaded via the *Persister* module. The *Persister's* chunk size is set to one hour, to make it comparable to loading from miniseed files. In iteration 1 the *Persister* loads the data via the *Seismic* data access unit and stores in the persist storage. In iteration 2 the *Persister* loads directly from the persist storage.
3. **Waveform+Persist+SamplePersist**: The same as (2) but a least-recently-used cache is added, which can cache more than two chunks in memory.
4. **Waveform+Persist+SamplePersist**: The same as (2) but another *Persister* is added to the chain which stores each 30 second sample individually.
5. **Waveform+Persist+SamplePersist (raw)**: The same as (3) but the data is loaded without coordinates (bypassing xarray).

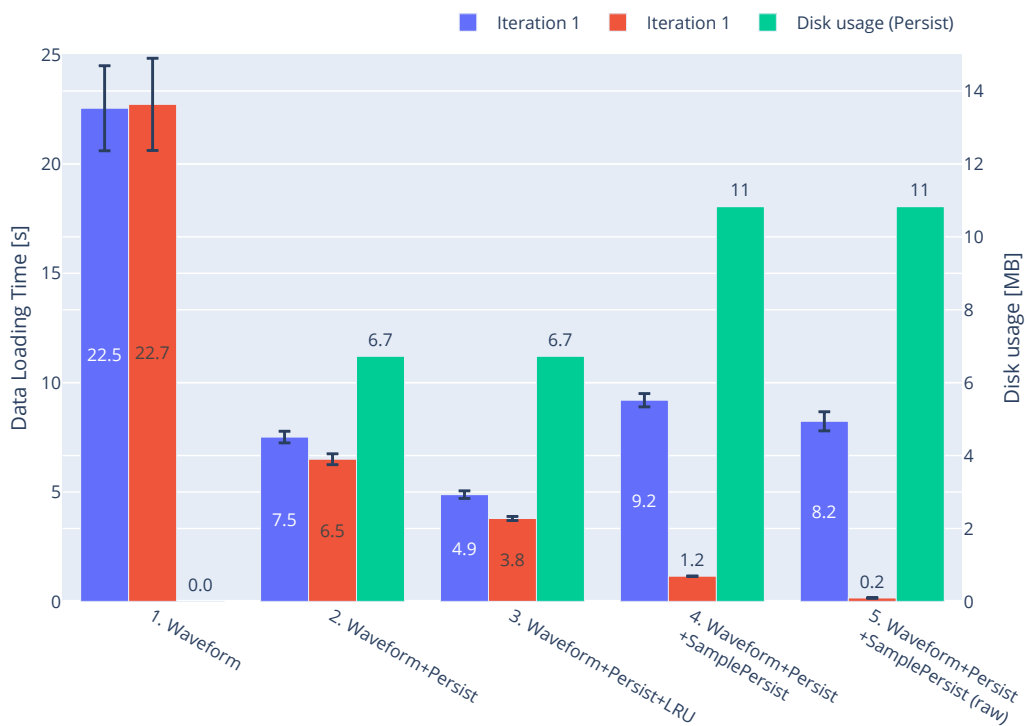


Figure 5.11 Data loading times and disk usage for benchmarks resembling a machine learning scenario in which data is randomly loaded from disk. For both iterations the results are given as mean and standard deviation (black bars) computed over 10 experiment repetitions. The disk usage is computed after the first iteration.

Figure 5.11 indicates that loading data using the *Persister* provides a huge improvement of up to about 19x in a scenario where all coordinates must be available (Benchmark 4). If only the raw data is required (Benchmark 5) overall improvement with regard to the baseline is more than about 100x. The figure also highlights the tradeoff between data loading times and disk usage. Each additional persister increases the data that needs to be stored on disk in addition to the original data. However, disk capacity is usually not the bottleneck in computing system due to cheap mass storage options.

5.4.1.1 Downsampling

The ability to plot long-term datasets relies heavily on the available compute power. Even with a low sampling rates of 100 Hz, one year of single-channel seismic data contains more than 3 billion samples. Modern plotting libraries [Plo21, Bok21] are able to display millions of data points, by using GPU acceleration [Web11] or pre-computed plots [Dat21]. However, from a certain data volume these techniques are not sufficient anymore and a data reduction before plotting is required. Moreover, even if we leave out the performance issues of plotting, just loading more than a billion data samples is a significant performance factor.

When plotting many points, a phenomena called overplotting can occur, meaning that close points or lines of the same color are indistinguishable because they overlap. We can use this fact to our advantages when plotting long-term dataset. On a large scale, overplotting occurs quite frequently and not all data points are required to give the correct visual impression. Therefore, the signal may be downsampled before plotting.

A simple method to downsample is to keep only every n -th sample (n is the downsampling rate). However, this may lead to data artifacts which significantly affect the accurate data representation as illustrated in Figure 5.12. Here, the downsampled version is non-representative of most of the original data neither in amplitude nor in frequency. In this subsection, we list two approaches which can be used to produce faithful visualizations when datasets need to be plotted on a large scale, namely MinMax Downsampling and Largest-Triangle-Three-Buckets [Ste13].

Largest-Triangle-Three-Buckets (LTTB) method selects the first and last sample of the signal and subdivides the rest of the signal into equal-sized bins (or buckets) of length equal to the downsampling rate. For every bin every point is ranked by computing "the area of the triangle it forms with the selected point in the last [bin] and the average point in the next [bin]" [Ste13]. The point with the highest rank is selected. The result of the LTTB algorithm applied to a chirp signal is illustrated in Figure 5.13. The large-scale plot indicates that LTTB gives a correct visual impression for low-frequency events but fails for high frequencies. Therefore, it is most suitable for slowly changing signals, for example rock temperature or weather data.

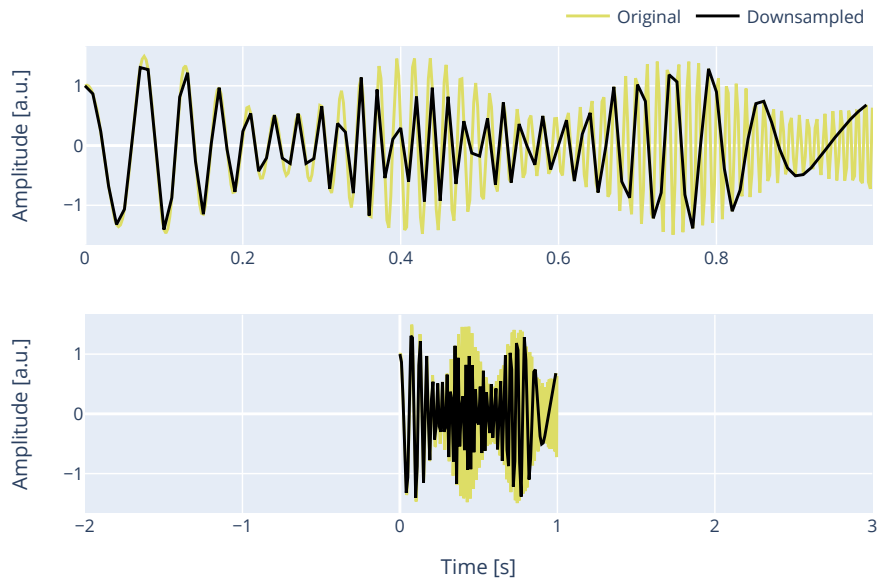


Figure 5.12 Plot of a chirp (a signal with increasing frequency) in yellow and the downsampled version obtained by using every 8th sample (black). The same signal is plotted on a small-scale (top) and large-scale (below). Downsampling artifacts give a false intuition of the signals amplitude as well as frequency

MinMax Downsampling subdivides the signal into equal-sized bins of length equal to twice the downsampling rate. For each bin the minimum and maximum samples are chosen. The result of the MinMax algorithm applied to a chirp signal is illustrated in Figure 5.14. The large-scale plot indicates that MinMax Downsampling gives a correct visual impression for high-frequencies but fails for low frequencies. Therefore, it is suited for waveform data, like seismic or audio.

5.4.2 Permavis

Permavis is a web frontend using *foReal* as data backend. It is designed to plot seismic data, time-lapse images, annotations and time series data from CSV or PermaSense . Moreover, it features a tool to load, edit and export annotation files, as well as drawing new annotations inside any plot. The frontend can be run on a local machine or deployed as a webservice. The web service features multi-scale, multi-modal data visualization capabilities, using downsampling (LTTB for time-series and MinMax for seismic waveforms) and persist to enable a responsive data exploration. Figure ?? displays a screenshot of a joint visualization of spectrogram, waveform, rain and wind.

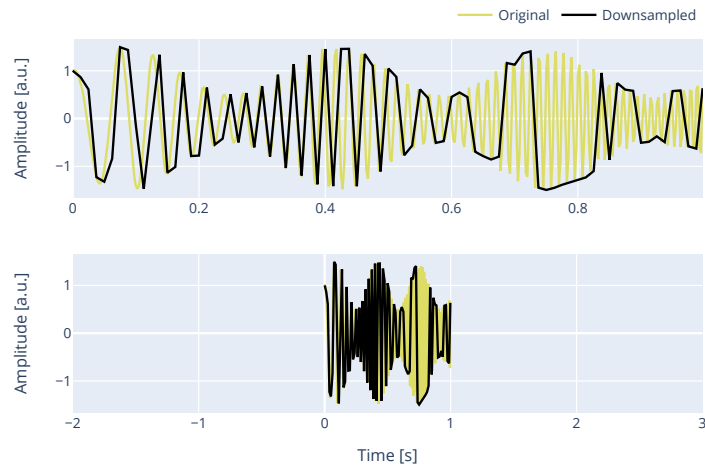


Figure 5.13 Plot of a chirp (yellow) and the downsampled version (black) obtained by using the LTTB method with downsampling rate of 8. The same signal is plotted on a small-scale (top) and large-scale (below). The large-scale plot indicates that LTTB gives a correct visual impression for low-frequencies but fails for high frequencies.

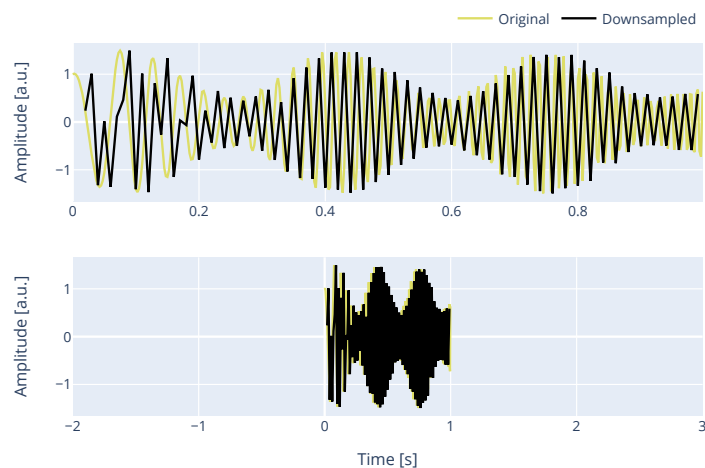


Figure 5.14 Plot of a chirp (yellow) and the downsampled version (black) obtained by using the MinMax method with downsampling rate of 8. The same signal is plotted on a small-scale (top) and large-scale (below). The large-scale plot indicates that MinMax Downsampling gives a correct visual impression for high-frequencies but fails for low frequencies.



Figure 5.15 Joint visualization of seismic spectrogram, seismic waveform, rain and wind data of the Matterhorn deployment for a time period of **more than a year**. Additionally, the annotations of an event catalog and wind data from a nearby weather station (ZER4 by SLF) are plotted.



Figure 5.16 Joint visualization of seismic spectrogram, seismic waveform, rain and wind data of the Matterhorn deployment for a time period of **multiple days**. Additionally, the annotations of an event catalog and wind data from a nearby weather station (ZER4 by SLF) are plotted.

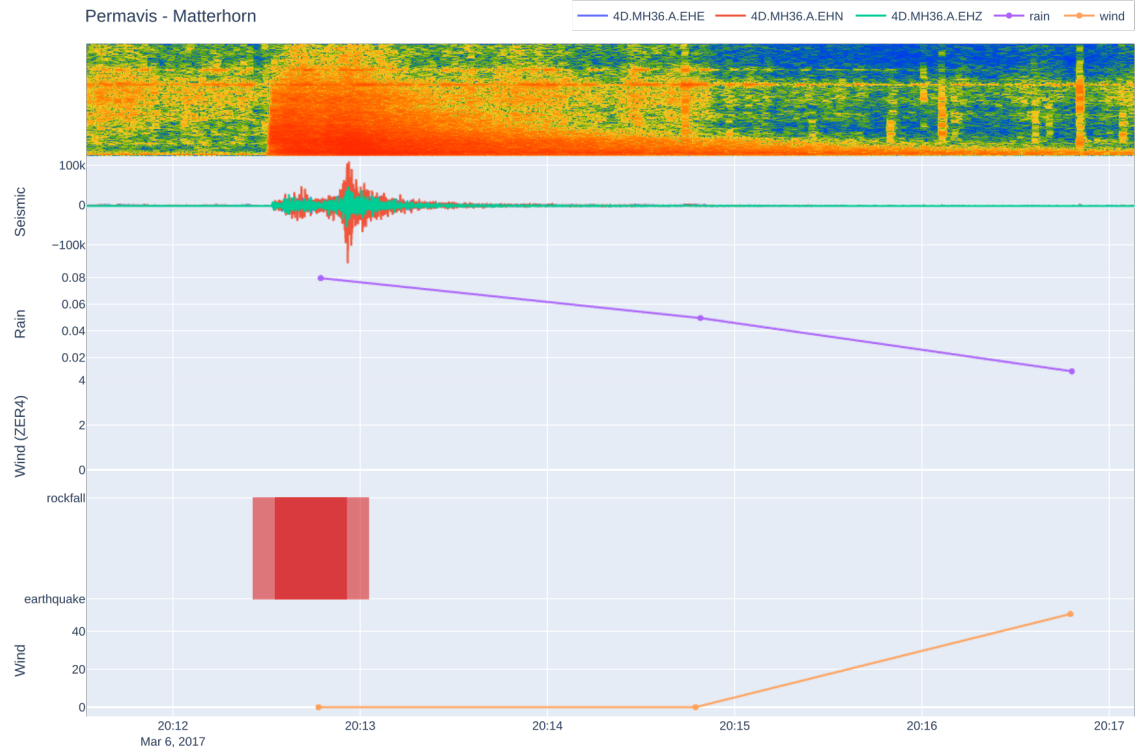


Figure 5.17 Joint visualization of seismic spectrogram, seismic waveform, rain and wind data of the Matterhorn deployment for a time period of **multiple minutes**. Additionally, the annotations of an event catalog and wind data from a nearby weather station (ZER4) are plotted.

5.5 Interactions

foReal is designed to minimize the time to access real-world datasets for model development and algorithm application. Ideally, the data science community can benefit from real-world datasets and the geoscience community from novel algorithms. Authorities can have almost immediate access to intelligible expert knowledge in case of emergencies supporting fast, fact-based decision making. If data is augmented with risk assessment it can be advantageous for other parties, for example for re-insurances and long-term policy and mitigation efforts. The general public can be invited to interact with their environment through data analysis and the contribution of observations (crowd-sourcing). Since they are exposed to the same tooling experts use, they get access to in-depth knowledge while data annotations (by experts) facilitate the understanding of natural phenomena. For local communities affected by environmental changes it is important that the gained insights about their habitat is fed back in an accessible manner no matter where or by whom these insights were generated.

One goal of this dissertation is to raise awareness, provide insight and foster interdisciplinary knowledge exchange in the focus area environmental monitoring and climate change and highlight that public awareness and collaboration is a key to reduce the impact of climate change. Thus, the presented toolsuite supports dynamic interaction between data, expert knowledge and machine learning technology.

Throughout the dissertation interaction with the public, experts, companies and artists was sought and fostered. These interactions include presentations at conferences, for high school and ETH alumni. Moreover, segments of seismic data from the Matterhorn were converted to soundfiles (freely accessible [Sou]). These soundfiles were presented at the "ZHdK Environmental Listening Session: Unknown Soundscapes: Rivers, Soils, Rocks" and are featured in the music project "Glacier Music II" by Anushka Chkheidze, Eto Gelashvili, Hayk Karoyi, Lillevan, Robert Lippok.

While this chapter highlights the conceptual benefits of our framework, it's usefulness will only show in practice. Thus, we have suited our action to our words by organizing a Hackathon on Permafrost in collaboration with Microsoft Switzerland which was staged at ETH Zurich November 28-30, 2019. During the event the participants had the chance to use a preliminary version of the *foReal* framework for data access and algorithm development. 87 participants from academia and industry registered out of which roughly 60 participated in the three-day event. The participant list included geoscientist, data scientists and participants with various other backgrounds. The event was structured into introductory sessions, technology and geo-science breakout sessions and group-work. Lorenz Meier (Geopraevent AG) gave a special industry keynote with case studies on natural hazard monitoring projects. Experts from Swiss Re and Microsoft supported the participants on topics of data analytics and application building. A total of 14 final solution templates were submitted by

individual groups spanning time-series data analysis, prediction and simulation tools, participatory apps as well as machine-learning based anomaly detection.

5.6 Conclusion

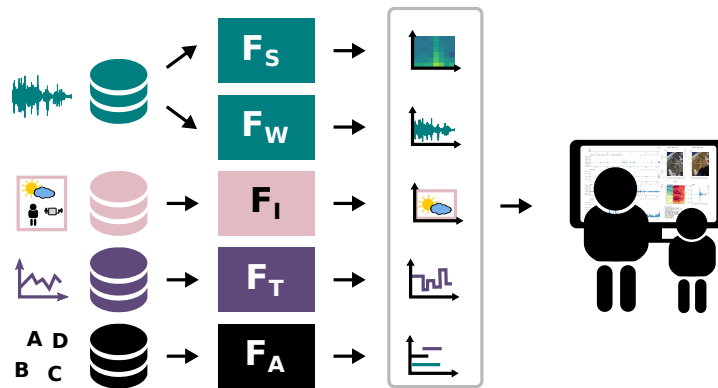


Figure 5.18 Data types and preprocessing used in this chapter. Expert-based information extraction from seismic data, images, time-series data and annotations using data specific frontends (F_W, F_S, F_T, F_A) and data specific plot types. The data is jointly analyzed by one or many experts.

In this chapter, we presented a software processing and analysis framework for heterogenous wireless sensor network deployments. A common online analysis platform can enable fact-based and fast-paced collaboration and encourage spontaneous, short-term collaboration between experts of different domains combining and complementing their strengths for a qualitative, comprehensive analysis. Moreover, it allows to establish crucial understanding between short-term effects and longer-term process models. While the understanding of these effects can be established by human experts using few examples, the application to long-term data streams must be automated through intelligent algorithms. We have shown how we can optimize data access, computation and visualization for large-scale datasets. The presented framework places at its core the principle of an ever-evolving dataset with its continuous data generation, dataset refinement and error-correction. Its focus on reproducibility makes the data accessible, persistent and tangible for the next research project, the next expert continuing the research cycle.

5.A Appendix

Contributions: Matthias Meyer developed the concept and discussed it with Jan Beutel, Lothar Thiele, Samuel Weber, Andreas Biri, Feiyu Jia, Tobias Kuonen, Carsten Barth. Matthias Meyer developed the framework's code with contributions from Samuel Weber, Andreas Biri, Reto da Forno. Matthias Meyer, Reto da Forno, Feiyu Jia and Tobias Kuonen developed the permavis web frontend. Matthias Meyer prepared and performed the experiments and evaluated the results with Reto Da Forno. Matthias Meyer prepared the manuscript.

 Project Webpage matthiasmeyer.xyz/research/foreal/

Related Publications

The *foReal* framework was used for running the experiments in Section 4.2 and its related publication.

The following student projects are related to this chapter.

Dynamic Visualization of Geophysical Data

Feiyu Jia

Semester thesis, 2019

Visual analysis and comparison of seismic events

Tobias Kuonen

Semester thesis, 2019

6

Conclusions and Outlook

Addressing the impact of global warming requires thorough analysis supported by evidence from in-situ environmental measurements. A qualitative analysis requires foremost a good data basis. Unfortunately, the data quality originating from environmental monitoring setups is often affected by natural uncertainties, technological and human errors. These errors manifest themselves on multiple stages during data acquisition and analysis. In this dissertation, we have presented tools and techniques to harness such erroneous environmental data. In particular, we have focused on harnessing data collected to understand the geophysical processes leading to rockfalls in steep terrain and datasets meant to quantify the impact of climate change on high-alpine permafrost. Moreover, we have presented a methodology for automated analysis of long-term, heterogeneous datasets which includes experts as well as machine learning. In this context, we have tackled the issue of noisy, partially labeled and fragmented datasets by using semi-supervised learning methods which integrate system context information. Additionally, a model for detection of mountaineers in the hazard zone was optimized for event-triggered, low-power geophones. An array of these geophones was deployed to evaluate their usefulness as a natural hazard warning system. Finally, a data analysis and visualization platform was developed enabling interactions between experts of various domains and the public.

6.1 Contributions

A methodology for working with noisy data (Chapter 2): We have presented a methodology to evaluate the impact of external influences on micro-seismic measurements using additional sensors and external information. Based on data from a real-world wireless sensor network deployment at Matterhorn, Switzerland, we have demonstrated how such information is useful to isolate

negative external influences, excluding them from further analysis. Moreover, we have designed a machine learning based mountaineer detector obtaining an error rate of only 0.96 % (F1 score: 0.9167) on micro-seismic signals and a mountaineer detector ensemble obtaining an error rate of 0.79 % (F1 score: 0.9383) on images and micro-seismic data. Furthermore, we have shown that using only a simple seismic event detector for analysis may lead to false assumptions, since time periods with mountaineer activity have a approximately 9x higher event rate and approximately 25% of all detected events are due to mountaineer interference.

Edge computing (Chapter 3): We have demonstrated a design for an event-driven, low-power geophone array, optimized for natural hazard warning and detection of human presence in a hazard zone. Our study finds that using analog triggering and on-device signal characterization can significantly extend the expected lifetime of the system. Moreover, we developed a footstep detector using machine learning and optimized it for on-device inference on a low-power microprocessor with a limited SRAM of only 320 kB. We applied architectural optimizations to reduce the memory requirement of an reference event detection model by a factor of 515. In addition we applied network quantization to reduce the parameter's memory requirement by another factor of 4. Additionally, we pipelined a convolutional neural network, reducing the inference-time and the inference-memory and keeping them independent of the temporal size of the networks input.

Domain expertise and machine learning (Chapter 4): Our contributions include methods to enhance domain expertise with machine learning by using unsupervised and semi-supervised deep representation learning. Moreover, we have demonstrated that domain expertise can improve machine-learning-based information extraction by incorporating system context information. For this purpose we leveraged in-batch similarity optimization in combination with an information graph producing representation useful for classification and clustering. We enhanced this optimization method by unifying data and annotation representations: annotations are represented as data in the embedding space. In a scenario with limited annotations and diverging sensor characteristics we demonstrated increased robustness across sensors.

Data analysis and visualization (Chapter 5): Finally, we developed and published the data analytics and visualization platform *foReal* which allows to combine data from different sources. The platform can be used to process and analyse long-term, large-scale environmental datasets while providing robustness against data corruption, missing data and misconfigurations during data processing as well as misinterpretations during experiment design and analysis. The system is designed to harmonize machine learning and expert knowledge and the tooling enables fact-based and fast-paced collaboration combining and complementing expert knowledge from various backgrounds. The web-based visualization software allows to interact with the public, opening up research results to a broad audience.

6.2 Future Developments

Real-world applications: The tools presented in this thesis have already been used to analyze real-world data. In general, they were designed to be applicable to other domains involving data from wireless sensor networks. Other application scenarios include debris flow warning [CWW⁺21], pollen sensing [CMTS20] or air pollution monitoring [MZT18, CHZT20]. Especially the *foReal* data analysis framework may be useful for other long-term datasets which require an integration of machine learning and expert knowledge. The machine learning methods presented in this dissertation may be readily transferred to audio event detection applications requiring only parameter changes.

Common toolkit for active learning: Errors during data analysis are among others introduced by non-unified set of analysis tools. To reduce such errors machine learning and data verification should be tightly coupled. The *foReal* framework (Chapter 5) already provides such options to some extent. As an addition, it may be integrated with the learning framework presented in Chapter 4 resulting in an active learning framework. Here, visualization of the embeddings and correlating clusters of data points in the embedding space with their representation in the original data space may help to develop an intuition about the event characteristics.

Embedded Learning: Machine learning on embedded devices are restricted by memory and computational power. We have demonstrated in Chapter 3 that computing the forward pass of a convolutional neural network is feasible on an embedded device by applying model optimization and re-using memory. This strategy cannot be applied to on-device learning since most of the intermediate values of the forward pass need to be kept in memory during the training phase. Therefore, simple models are preferred for on-device training which however limits the classification accuracy.

The method presented in Chapter 4 splits the model into a complex encoder and simple decoder which can be trained independently. This method provides options to tackle the issues of on-device learning. The encoder can be pre-trained on a compute cluster using all available data (annotated and non-annotated). Then, on the embedded device only the forward pass of the encoder needs to be computed. The decoder can be pre-trained on the server and then be fine-tuned on-device. Moreover, this method provides options to run a complex decoder on the server and a simple decoder on-device - a tradeoff between transmission cost and accuracy. The embedded devices transmits only an intermediate representation produced by the encoder or the classification result. The original data of specific events can be requested on-demand. Here, the intermediate representations or the on-device classification can give an intuition if the data of a particular segment is of interest at all (for example if only noise was classified on device, the data does not have to be transmitted).

Bibliography

- [AAB⁺15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015.
- [AAC⁺10] D. Amitrano, M. Arattano, M. Chiarle, G. Mortara, C. Occhiena, M. Pirulli, and C. Scavia. Microseismic activity analysis for the study of the rupture mechanisms in unstable rock masses. *Natural Hazards and Earth System Sciences*, 10(4):831–841, April 2010.
- [AB14] A. Aguiar and G. Beroza. PageRank for earthquakes. *Seismological Research Letters*, 85:344–350, March 2014.
- [ACO11] S. K. Allen, S. C. Cox, and I. F. Owens. Rock avalanches and other landslides in the central Southern Alps of New Zealand: A regional study considering possible climate change impacts. *Landslides*, 8(1):33–48, March 2011.
- [ACRB18] R. Andri, L. Cavigelli, D. Rossi, and L. Benini. YodaNN: An Architecture for Ultralow Power Binary-Weight CNN Acceleration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):48–60, January 2018.
- [AGG12] D. Amitrano, S. Gruber, and L. Girard. Evidence of frost-cracking inferred from acoustic emissions in a high-alpine rock-wall. *Earth and Planetary Science Letters*, 341-344:86–93, August 2012.
- [AGS05] D. Amitrano, J. R. Grasso, and G. Senfaute. Seismic Precursory Patterns before a Cliff Collapse and Critical Point Phenomena. *Geophysical Research Letters*, 32(8):L08314, April 2005.
- [All78] R. V. Allen. Automatic earthquake recognition and timing from single traces. *Bulletin of the Seismological Society of America*, 68(5):1521–1532, October 1978.
- [Ama] Amazon SageMaker Ground Truth | AWS. <https://aws.amazon.com/sagemaker/groundtruth/>.
- [AMK18] S. Anchal, B. Mukhopadhyay, and S. Kar. UREDT: Unsupervised Learning Based Real-Time Footfall Event Detection Technique in Seismic Signal. *IEEE Sensors Letters*, 2(1):1–4, March 2018.
- [Apa] Apache Spark™ - Unified Analytics Engine for Big Data. <https://spark.apache.org/>.
- [Arc] Arclink WebDC3 Web Interface at SED-ETHZ. <http://arclink.ethz.ch/webinterface/>.

- [ArMJ⁺14] O. Abdel-Hamid, A. r Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, October 2014.
- [ASC⁺10] K. Aberer, S. Sathe, D. Chakraborty, A. Martinoli, G. Barrenetxea, B. Faltings, and L. Thiele. OpenSense: Open community driven sensing of environment. In *Proceedings of the ACM SIGSPATIAL International Workshop on GeoStreaming, IWGS '10*, pages 39–42, New York, NY, USA, November 2010. Association for Computing Machinery.
- [BAW⁺15] T. C. Bartholomaus, J. M. Amundson, J. I. Walter, S. O’Neel, M. E. West, and C. F. Larsen. Subglacial discharge at tidewater glaciers revealed by seismic tremor. *Geophysical Research Letters*, 42(15):6391–6398, August 2015.
- [BB18] J. Binas and Y. Bengio. Low-Memory Convolutional Neural Networks through Incremental Depth-First Processing. *arXiv:1804.10727 [cs]*, April 2018.
- [BBF⁺11a] J. Beutel, B. Buchli, F. Ferrari, M. Keller, L. Thiele, and M. Zimmerling. X-Sense: Sensing in Extreme Environments. *Proceedings of Design, Automation and Test in Europe (DATE 2011)*, pages 1460–1465, March 2011.
- [BBF⁺11b] J. Beutel, B. Buchli, F. Ferrari, M. Keller, M. Zimmerling, and L. Thiele. X-SENSE: Sensing in extreme environments. In *2011 Design, Automation Test in Europe*, pages 1–6, March 2011.
- [BBS08] J. R. Brown, G. C. Beroza, and D. R. Shelly. An autocorrelation method to detect low frequency earthquakes within tremor. *Geophysical Research Letters*, 35(16), August 2008.
- [BC20] E. K. Bessette-Kirton and J. A. Coe. A 36-Year Record of Rock Avalanches in the Saint Elias Mountains of Alaska, With Implications for Future Hazards. *Frontiers in Earth Science*, 8:293, 2020.
- [BGH⁺09a] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yucel. PermaDAQ: A scientific instrument for precision sensing and data recovery in environmental extremes. 2009.
- [BGH⁺09b] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yucel. PermaDAQ: A Scientific Instrument for Precision Sensing and Data Recovery in Environmental Extremes. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, IPSN '09*, pages 265–276. IEEE Computer Society, 2009.
- [BHB19] P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning Representations by Maximizing Mutual Information Across Views. *arXiv:1906.00910 [cs, stat]*, June 2019.
- [BHK20] A. Blum, J. Hopcroft, and R. Kannan. *Foundations of Data Science*. Cambridge University Press, first edition, January 2020.
- [BMC⁺11] D. T. Blumstein, D. J. Mennill, P. Clemins, L. Girod, K. Yao, G. Patricelli, J. L. Deppe, A. H. Krakauer, C. Clark, K. A. Cortopassi, S. F. Hanser, B. McCowan, A. M. Ali, and A. N. G. Kirschel. Acoustic monitoring in terrestrial environments using microphone arrays: Applications, technological considerations and prospectus. *Journal of Applied Ecology*, 48(3):758–767, June 2011.
- [BMMF12] J. Burjánek, J. R. Moore, F. X. Y. Molina, and D. Fäh. Instrumental evidence of normal mode rock slope vibration. *Geophysical Journal International*, 188(2):559–569, February 2012.

- [BNK16] A. Butterfield, G. E. Ngondi, and A. Kerr, editors. *A Dictionary of Computer Science*. Oxford University Press, January 2016.
- [Bok21] Bokeh. Bokeh, August 2021.
- [Bre01] L. Breiman. Random forests. *Machine Learning*, 2001.
- [BSN⁺19] B. K. Biskaborn, S. L. Smith, J. Noetzli, H. Matthes, G. Vieira, D. A. Streletskiy, P. Schoeneich, V. E. Romanovsky, A. G. Lewkowicz, A. Abramov, M. Allard, J. Boike, W. L. Cable, H. H. Christiansen, R. Delaloye, B. Diekmann, D. Drozdov, B. Etzelmüller, G. Grosse, M. Guglielmin, T. Ingeman-Nielsen, K. Isaksen, M. Ishikawa, M. Johansson, H. Johannsson, A. Joo, D. Kaverin, A. Kholodov, P. Konstantinov, T. Kröger, C. Lambiel, J.-P. Lanckman, D. Luo, G. Malkova, I. Meiklejohn, N. Moskalenko, M. Oliva, M. Phillips, M. Ramos, A. B. K. Sannel, D. Sergeev, C. Seybold, P. Skryabin, A. Vasiliev, Q. Wu, K. Yoshikawa, M. Zheleznyak, and H. Lantuit. Permafrost is warming at a global scale. *Nature Communications*, 10(1):264, January 2019.
- [BTF⁺19] J. Beutel, R. Trüb, R. D. Forno, M. Wegmann, T. Gsell, R. Jacob, M. Keller, F. Sutton, and L. Thiele. The dual processor platform architecture: Demo abstract. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks, IPSN '19*, pages 335–336, New York, NY, USA, April 2019. Association for Computing Machinery.
- [BvW07] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-low power data gathering in sensor networks. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks, IPSN '07*, pages 450–459, New York, NY, USA, April 2007. Association for Computing Machinery.
- [CB15] F. Conti and L. Benini. A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 683–688. EDA Consortium, 2015.
- [CBM07] M. Cristani, M. Bicego, and V. Murino. Audio-Visual Event Recognition in Surveillance Video Sequences. *IEEE Transactions on Multimedia*, 9(2):257–267, February 2007.
- [CCVB18] C. Colombero, C. Comina, S. Vinciguerra, and P. M. Benson. Microseismicity of an Unstable Rock Mass: From Field Monitoring to Laboratory Testing. *Journal of Geophysical Research: Solid Earth*, February 2018.
- [CF13] S. A. Crate and A. N. Fedorov. A Methodological Model for Exchanging Local and Scientific Climate Change Knowledge in Northeastern Siberia. *ARCTIC*, 66(3):338–350, September 2013.
- [CFCS17] K. Choi, G. Fazekas, K. Cho, and M. Sandler. A Comparison of Audio Signal Preprocessing Methods for Deep Neural Networks on Music Tagging. *arXiv:1709.01922 [cs]*, September 2017.
- [CGM⁺15] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini. Origami: A convolutional network accelerator. In *Proceedings of the 25th Edition on Great Lakes Symposium on VLSI, GLSVLSI '15*, pages 199–204, New York, NY, USA, 2015. ACM.
- [CHHV15] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen. Polyphonic Sound Event Detection Using Multi Label Deep Neural Networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, July 2015.
- [Cho15] F. Chollet. Keras, 2015.

- [CHZT20] Y. Cheng, X. He, Z. Zhou, and L. Thiele. MapTransfer : Urban Air Quality Map Generation for Downscaled Sensor Deployments. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 14–26, April 2020.
- [CKES17] Y. Chen, T. Krishna, J. S. Emer, and V. Sze. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, January 2017.
- [CKNH20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*, February 2020.
- [CMTS20] N. Cao, M. Meyer, L. Thiele, and O. Saukh. Automated Pollen Detection with an Affordable Technology. In *Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks on Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks, EWSN '20*, pages 108–119, USA, February 2020. Junction Publishing.
- [CPH⁺17] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, T. Virtanen, E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen. Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(6):1291–1303, June 2017.
- [CWW⁺21] M. Chmiel, F. Walter, M. Wenner, Z. Zhang, B. W. McArdell, and C. Hibert. Machine Learning Improves Debris Flow Warning. *Geophysical Research Letters*, 48(3):e2020GL090874, 2021.
- [Dasa] Dask documentation - Comparison to Spark. <https://docs.dask.org/en/latest/spark.html>.
- [Dasb] Dask: Scalable analytics in Python. <https://dask.org/>.
- [Dat21] Datashader. HoloViz, August 2021.
- [DDS⁺09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [DED⁺16] S. Dickerson-Lange, K. Eitel, L. Dorsey, T. Link, and J. Lundquist. Challenges and successes in engaging citizen scientists to observe snow cover: From public engagement to an educational collaboration. *Journal of Science Communication*, 15(01):A01, January 2016.
- [DG08] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, January 2008.
- [DH04] M. F. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.
- [DMH⁺16] F. Dammeier, J. R. Moore, C. Hammer, F. Haslinger, and S. Loew. Automatic detection of alpine rockslides in continuous seismic data using hidden Markov models. *Journal of Geophysical Research: Earth Surface*, 2016.
- [DS14] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968, May 2014.
- [DT21] S. Draskovic and L. Thiele. Optimal Power Management for Energy Harvesting Systems with A Backup Power Source. In *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–9, June 2021.

- [EFKN15] M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani. Exploiting spectro-temporal locality in deep learning based acoustic event detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):26, December 2015.
- [ELBA17] E. P. S. Eibl, I. Lokmer, C. J. Bean, and E. Akerlie. Helicopter location and tracking using seismometer recordings. *Geophysical Journal International*, 209(2):901–908, May 2017.
- [FFBV18] J. Faillettaz, M. Funk, J. Beutel, and A. Vieli. Co-detection of micro seismic activity as early warning of gravitational slope failure. *Natural Hazards and Earth System Sciences Discussions*, 2018:1–17, 2018.
- [FFP06] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, April 2006.
- [FGL16] C. Finn, I. Goodfellow, and S. Levine. Unsupervised Learning for Physical Interaction through Video Prediction. *Advances in Neural Information Processing Systems 29*, pages 64–72, 2016.
- [FGO⁺10] F. Flammini, A. Gaglione, F. Ottello, A. Pappalardo, C. Pragliola, and A. Tedesco. Towards Wireless Sensor Networks for railway infrastructure monitoring. In *Railway and Ship Propulsion Electrical Systems for Aircraft*, pages 1–6, October 2010.
- [FKHN06] L. Fischer, A. Kääh, C. Huggel, and J. Noetzli. Geology, glacier retreat and permafrost degradation as controlling factors of slope instabilities in a high-mountain rock wall: The Monte Rosa east face. *Natural Hazards and Earth System Sciences*, 6(5):761–772, September 2006.
- [FOM⁺20] E. Fonseca, D. Ortego, K. McGuinness, N. E. O’Connor, and X. Serra. Unsupervised Contrastive Learning of Sound Event Representations. *arXiv:2011.07616 [cs, eess]*, November 2020.
- [FOR16] J. Faillettaz, D. Or, and I. Reiweger. Codetection of Acoustic Emissions during Failure of Heterogeneous Media: New Perspectives for Natural Hazard Early Warning: CODETECTION OF ACOUSTIC EMISSIONS. *Geophysical Research Letters*, 43(3):1075–1083, February 2016.
- [FPH⁺12] L. Fischer, R. S. Purves, C. Huggel, J. Noetzli, and W. Haeberli. On the influence of topographic, geological and cryospheric factors on rock avalanches and rockfalls in high-mountain areas. *Natural Hazards and Earth System Sciences*, 12(1):241–254, January 2012.
- [FZMT12] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, SenSys ’12*, pages 1–14, New York, NY, USA, November 2012. Association for Computing Machinery.
- [FZTS11] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 73–84, April 2011.
- [GAC05] T. Glade, M. Anderson, and M. J. Crozier. *Landslide Hazard and Risk*. John Wiley & Sons, Ltd, Chichester, West Sussex, England, February 2005.
- [GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, 2016.

- [GBG⁺12] L. Girard, J. Beutel, S. Gruber, J. Hunziker, R. Lim, and S. Weber. A custom acoustic emission monitoring system for harsh environments: Application to freezing-induced damage in alpine rock walls. *Geoscientific Instrumentation, Methods and Data Systems*, 1(2):155–167, 2012.
- [GEMH15] V. S. Gischig, E. Eberhardt, J. R. Moore, and O. Hungr. On the seismic response of deep-seated rock slope instabilities — Insights from numerical modeling. *Engineering Geology*, 193:1–18, July 2015.
- [Geo18] Geometrics. Geode Exploration Seismograph Specification Sheet, 2018.
- [GH07] S. Gruber and W. Haeberli. Permafrost in steep bedrock slopes and its temperature-related destabilization following climate change. *Journal of Geophysical Research*, 112(F2):F02S18, June 2007.
- [GHH04] S. Gruber, M. Hoelzle, and W. Haeberli. Rock-wall temperatures in the Alps: Modelling their topographic distribution and regional differences. *Permafrost and Periglacial Processes*, 15(3):299–307, July 2004.
- [GKK⁺04] S. Gruber, L. King, T. Kohl, T. Herz, W. Haeberli, and M. Hoelzle. Interpretation of geothermal profiles perturbed by topography: The alpine permafrost boreholes at Stockhorn Plateau, Switzerland. *Permafrost and Periglacial Processes*, 15(4):349–357, October 2004.
- [GO08] C. U. Grosse and M. Ohtsu, editors. *Acoustic Emission Testing*. Springer-Verlag, Berlin Heidelberg, 2008.
- [GR06] S. J. Gibbons and F. Ringdal. The detection of low magnitude seismic events using array-based waveform correlation. *Geophysical Journal International*, 165(1):149–166, April 2006.
- [GSA⁺20] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. *arXiv:2006.07733 [cs, stat]*, June 2020.
- [GSS⁺17] A. Gomez, L. Sigrist, T. Schalch, L. Benini, and L. Thiele. Efficient, Long-Term Logging of Rich Data Sensors Using Transient Sensor Nodes. *ACM Transactions on Embedded Computing Systems*, 17(1):4:1–4:23, September 2017.
- [GTWX11] Y. Gui, Z.-g. Tao, C.-j. Wang, and X. Xie. Study on remote monitoring system for landslide hazard based on wireless sensor network and its application. *Journal of Coal Science and Engineering (China)*, 17(4):464–468, December 2011.
- [Har03] H. R. Hardy. *Acoustic Emission, Microseismic Activity*. Balkema, Lisse, 2003.
- [HBC⁺21] A. Hogan, E. Blomqvist, M. Cochez, C. d’Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. N. Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann. Knowledge Graphs. *arXiv:2003.02320 [cs]*, January 2021.
- [HCE⁺16] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson. CNN Architectures for Large-Scale Audio Classification. *arXiv: 1609.09430*, September 2016.
- [HFL⁺18] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv:1808.06670 [cs, stat]*, August 2018.

- [HFW⁺20] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.
- [HH17] S. Hoyer and J. Hamman. Xarray: N-D labeled Arrays and Datasets in Python. *Journal of Open Research Software*, 5(1):10, April 2017.
- [HHVH⁺18] M. Heck, C. Hammer, A. Van Herwijnen, J. Schweizer, and D. Fäh. Automatic detection of snow avalanches in continuous seismic data using hidden Markov models. *Natural Hazards and Earth System Sciences*, 18(1):383–396, 2018.
- [HK15] Y.-C. Hsu and Z. Kira. Neural network-based clustering using pairwise constraints. *arXiv: 1511.06321*, November 2015.
- [HL09] P. T. P. Honglak Lee. Unsupervised feature learning for audio classification using convolutional deep belief networks. *NIPS*, pages 1096–1104, 2009.
- [HOF13] C. Hammer, M. Ohrnberger, and D. Fäh. Classifying seismic waveforms from scratch: A case study in the alpine environment. *Geophysical Journal International*, 2013.
- [HPM⁺17] C. Hibert, F. Provost, J. P. Malet, A. Maggi, A. Stumpf, and V. Ferrazzini. Automatic identification of rockfalls and volcano-tectonic earthquakes at the Piton de la Fournaise volcano using a Random Forest algorithm. *Journal of Volcanology and Geothermal Research*, 2017.
- [HRA⁺19] R. Hock, G. Rasul, C. Adler, B. Cáceres, S. Gruber, Y. Hirabayashi, M. Jackson, A. Kääb, S. Kang, S. Kutuzov, A. Milner, U. Molau, S. Morin, B. Orlove, and H. Steltzer. High Mountain Areas. *IPCC Special Report on the Ocean and Cryosphere in a Changing Climate [H.-O. Pörtner, D.C. Roberts, V. Masson-Delmotte, P. Zhai, M. Tignor, E. Poloczanska, K. Mintenbeck, A. Alegría, M. Nicolai, A. Okem, J. Petzold, B. Rama, N.M. Weyer (eds.)]*, 2019.
- [HS06] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.
- [HTB⁺08] A. Hasler, I. Talzi, J. Beutel, C. Tschudin, and S. Gruber. Wireless sensor networks in permafrost research: Concept, requirements, implementation, and challenges. 2008.
- [Hun00] H. P. Huntington. Using Traditional Ecological Knowledge in Science: Methods and Applications. *Ecological Applications*, 10(5):1270–1274, 2000.
- [HYA⁺18] K. Hegde, J. Yu, R. Agrawal, M. Yan, M. Pellauer, and C. W. Fletcher. UCNN: Exploiting Computational Reuse in Deep Neural Networks via Weight Repetition. *arXiv:1804.06508 [cs]*, April 2018.
- [HZC⁺17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]*, April 2017.
- [HZRS15] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015.
- [IG20] N. Inoue and K. Goto. Semi-Supervised Contrastive Learning with Generalized Contrastive Loss and Its Application to Speaker Recognition. In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1641–1646, December 2020.
- [IGM⁺12] E. Intrieri, G. Gigli, F. Mugnai, R. Fanti, and N. Casagli. Design and Implementation of a Landslide Early Warning System. *Engineering Geology*, 147-148:124–136, October 2012.

- [IS15] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, February 2015.
- [KASK16] Q. Kong, R. M. Allen, L. Schreier, and Y.-W. Kwon. MyShake: A smartphone seismic network for earthquake early warning and beyond. *Science Advances*, 2(2), 2016.
- [KB14] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014.
- [KBST12] M. Keller, J. Beutel, O. Saukh, and L. Thiele. Visualizing large sensor network data sets in space and time with vizzly. In *37th Annual IEEE Conference on Local Computer Networks - Workshops*, pages 925–933, October 2012.
- [KFG13] M. Krautblatter, D. Funk, and F. K. Günzel. Why permafrost rocks become unstable: A rock–ice–mechanical model in time and space. *Earth Surface Processes and Landforms*, 38(8):876–887, 2013.
- [KG17] K. V. Kislov and V. V. Gravirov. Use of artificial neural networks for classification of noisy seismic signals. *Seismic Instruments*, 53(1):87–101, January 2017.
- [KTKS16] T. Komatsu, T. Toizumi, R. Kondo, and Y. Senda. Acoustic Event Detection Method Using Semi-Supervised Non-Negative Matrix Factorization with a Mixture of Local Dictionaries. Technical report, DCASE2016 Challenge, September 2016.
- [KTW⁺20] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised Contrastive Learning. *arXiv:2004.11362 [cs, stat]*, December 2020.
- [Kuh55] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, March 1955.
- [KYB09] M. Keller, M. Yucel, and J. Beutel. High Resolution Imaging for Environmental Monitoring Applications. In *International Snow Science Workshop 2009: Programme and Abstracts*, pages 197–201, Davos, Switzerland, October 2009.
- [KYDH11] H. S. Kuyuk, E. Yildirim, E. Dogan, and G. Horasan. An unsupervised learning algorithm: Application to the discrimination of seismic events and quarry blasts in the vicinity of Istanbul. *Natural Hazards and Earth System Sciences*, 11(1):93–100, January 2011.
- [Lab] Labelbox. <https://labelbox.com/>.
- [LCC01] J. F. Labuz, S. Cattaneo, and L.-H. Chen. Acoustic emission at failure in quasi-brittle materials. *Construction and Building Materials*, 15(5-6):225–233, 2001.
- [LCY14] M. Lin, Q. Chen, and S. Yan. Network In Network. *ICLR: Conference Track*, 2014.
- [LJB11] C. Levy, D. Jongmans, and L. Baillet. Analysis of Seismic Signals Recorded on a Prone-to-Fall Rock Column (Vercors Massif, French Alps). *Geophysical Journal International*, 186(1):296–310, July 2011.
- [LMH⁺18] Z. Li, M.-A. Meier, E. Hauksson, Z. Zhan, and J. Andrews. Machine Learning Seismic Wave Discrimination: Application to Earthquake Early Warning. *Geophysical Research Letters*, 45(10):4773–4779, May 2018.

- [LMP⁺16] M. Lam, M. Mirshekari, S. Pan, P. Zhang, and H. Y. Noh. Robust Occupant Detection Through Step-Induced Floor Vibration by Incorporating Structural Characteristics. In M. Allen, R. L. Mayes, and D. Rixen, editors, *Dynamics of Coupled Structures, Volume 4*, Conference Proceedings of the Society for Experimental Mechanics Series, pages 357–367. Springer International Publishing, 2016.
- [MAP⁺15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015.
- [MBT17] M. Meyer, J. Beutel, and L. Thiele. Unsupervised Feature Learning for Audio Analysis. *arXiv:1712.03835 [cs]*, December 2017.
- [MCO12] G. Michlmayr, D. Cohen, and D. Or. Sources and characteristics of acoustic emissions from mechanically stressed geologic granular media — A review. *Earth-Science Reviews*, 112(3):97–114, May 2012.
- [Mer76] P. Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, 116:374–388, 1976.
- [MLB⁺17] A. Mathur, N. D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, and F. Kawsar. DeepEye: Resource Efficient Local Execution of Multiple Deep Vision Models Using Wearable Commodity Hardware. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '17*, pages 68–81. ACM, 2017.
- [MPC⁺10] L. Mottola, G. P. Picco, M. Ceriotti, Ş. Gună, and A. L. Murphy. Not all wireless sensor networks are created equal: A comparative study on tunnels. *ACM Transactions on Sensor Networks*, 7(2):15:1–15:33, September 2010.
- [MSC⁺19] M. Meredith, M. Sommerkorn, S. Cassotta, C. Derksen, A. Ekaykin, A. Hollowed, G. Kofinas, A. Mackintosh, J. Melbourne-Thomas, M. Muelbert, G. Ottersen, H. Pritchard, and E. Schuur. Polar Regions. *IPCC Special Report on the Ocean and Cryosphere in a Changing Climate [H.-O. Pörtner, D.C. Roberts, V. Masson-Delmotte, P. Zhai, M. Tignor, E. Poloczanska, K. Mintenbeck, A. Alegría, M. Nicolai, A. Okem, J. Petzold, B. Rama, N.M. Weyer (eds.)]*, 2019.
- [MTK17] B. McDanel, S. Teerapittayanon, and H. T. Kung. Embedded Binarized Neural Networks. *arXiv:1709.02260 [cs]*, September 2017.
- [MW18] M. Meyer and S. Weber. Code For Classifier Training And Evaluation Using The Micro-Seismic And Image Dataset Acquired At Matterhorn Hörnligrat, Switzerland. July 2018.
- [MWB⁺18] M. Meyer, S. Weber, J. Beutel, S. Gruber, T. Gsell, A. Hasler, and A. Vieli. Micro-Seismic And Image Dataset Acquired At Matterhorn Hörnligrat, Switzerland. July 2018.
- [MWBT19a] M. Meyer, S. Weber, J. Beutel, and L. Thiele. Systematic identification of external influences in multi-year microseismic recordings using convolutional neural networks. *Earth Surface Dynamics*, 7(1):171–190, February 2019.

- [MWBT19b] M. Meyer, S. Weber, J. Beutel, and L. Thiele. Systematic Identification of External Influences in Multi-Year Microseismic Recordings Using Convolutional Neural Networks. *Earth Surface Dynamics*, 7(1):171–190, 2019.
- [MZT18] B. Maag, Z. Zhou, and L. Thiele. A Survey on Sensor Calibration in Air Pollution Monitoring Deployments. *IEEE Internet of Things Journal*, 5(6):4857–4870, December 2018.
- [NAM21] C. G. Northcutt, A. Athalye, and J. Mueller. Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. *arXiv:2103.14749 [cs, stat]*, April 2021.
- [NGK⁺07] J. Noetzli, S. Gruber, T. Kohl, N. Salzmann, and W. Haeberli. Three-dimensional distribution and evolution of permafrost temperatures in idealized high-mountain topography. *Journal of Geophysical Research*, 112(F2):F02S13, May 2007.
- [NSQ⁺18] D. T. Nguyen, C. Song, Z. Qian, S. V. Krishnamurthy, E. J. M. Colbert, and P. McDaniel. lotSan: Fortifying the safety of IoT systems. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '18, pages 191–203, New York, NY, USA, December 2018. Association for Computing Machinery.
- [NXP16] NXP Semiconductors. *KV5x Data Sheet*, June 2016.
- [OCA⁺12] C. Occhiena, V. Coviello, M. Arattano, M. Chiarle, U. Morra di Cella, M. Pirulli, P. Pogliotti, and C. Scavia. Analysis of Microseismic Signals and Temperature Recordings for Rock Slope Stability Investigations in High Mountain Areas. *Natural Hazards and Earth System Sciences*, 12(7):2283–2298, July 2012.
- [OCB18] G. Olivier, J. Chaput, and B. Borchers. Using Supervised Machine Learning to Improve Active Source Signal Retrieval. *Seismological Research Letters*, 89(3):1023–1029, May 2018.
- [Pas18] A. Pasztor. *Event-Based Geophone Platform with Co-Detection*. PhD thesis, Computer Engineering and Networks Lab, ETH Zurich, May 2018.
- [Per] PermaSense :: GSN. <http://data.permasense.ch/>.
- [PGD18] T. Perol, M. Gharbi, and M. Denolle. Convolutional Neural Network for Earthquake Detection and Location. *Science Advances*, 4(2):e1700578, February 2018.
- [PGF18] P. Paitz, A. Gokhberg, and A. Fichtner. A Neural Network for Noise Correlation Classification. *Geophysical Journal International*, 212(2):1468–1474, February 2018.
- [PHM17] F. Provost, C. Hibert, and J. P. Malet. Automatic classification of endogenous landslide seismicity using the Random Forest supervised classifier. *Geophysical Research Letters*, 2017.
- [PHV16] G. Parascandolo, H. Huttunen, and T. Virtanen. Recurrent neural networks for polyphonic sound event detection in real life recordings. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6440–6444, March 2016.
- [Plo21] Plotly. Plotly, August 2021.
- [PMH⁺18] F. Provost, J.-P. Malet, C. Hibert, A. Helmstetter, M. Radiguet, D. Amitrano, N. Langet, E. Larose, C. Abancó, M. Hürlimann, T. Lebourg, C. Levy, G. Le Roy, P. Ulrich, M. Vidal, and B. Vial. Towards a standard typology of endogenous landslide seismic sources. *Earth Surface Dynamics*, 6(4):1059–1088, November 2018.

- [PRM⁺] H.-O. Pörtner, D. Roberts, V. Masson-Delmotte, P. Zhai, M. Tignor, E. Poloczanska, K. Mintenbeck, A. Alegría, M. Nicolai, A. Okem, J. Petzold, B. Rama, and N. W. (eds.). IPCC, 2019: IPCC Special Report on the Ocean and Cryosphere in a Changing Climate. Technical report, In press.
- [PVG⁺11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [PWQ⁺15] S. Pan, N. Wang, Y. Qian, I. Velibeyoglu, H. Y. Noh, and P. Zhang. Indoor Person Identification Through Footstep Induced Structural Vibration. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, HotMobile '15, pages 81–86. ACM, 2015.
- [RA17] A. Reynen and P. Audet. Supervised machine learning on a network scale: Application to seismic event classification and detection. *Geophysical Journal International*, 210(3):1394–1409, September 2017.
- [RD11] L. Raveland and P. Deline. Climate influence on rockfalls in high-Alpine steep rockwalls: The north side of the Aiguilles de Chamonix (Mont Blanc massif) since the end of the 'Little Ice Age'. *The Holocene*, 21(2):357–365, March 2011.
- [RDS⁺15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015.
- [RMB⁺14] A. E. Ruano, G. Madureira, O. Barros, H. R. Khosravani, M. G. Ruano, and P. M. Ferreira. Seismic detection using support vector machines. *Neurocomputing*, 135:273–283, July 2014.
- [RMH18] Z. E. Ross, M.-A. Meier, and E. Hauksson. P Wave Arrival Picking and First-Motion Polarity Determination With Deep Learning. *Journal of Geophysical Research: Solid Earth*, 123(6):5120–5129, June 2018.
- [RSG⁺21] E. Rusak, S. Schneider, P. Gehler, O. Bringmann, W. Brendel, and M. Bethge. Adapting ImageNet-scale models to complex distribution shifts with self-learning. *arXiv:2104.12928 [cs]*, April 2021.
- [SDBR14] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv:1412.6806 [cs]*, December 2014.
- [SDFBT17] F. Sutton, R. Da Forno, J. Beutel, and L. Thiele. BLITZ: A Network Architecture for Low Latency and Energy-Efficient Event-Triggered Wireless Communication. In *Proceedings of the 4th ACM Workshop on Hot Topics in Wireless*, HotWireless '17, pages 55–59. ACM, 2017.
- [SDFG⁺17] F. Sutton, R. Da Forno, D. Gschwend, T. Gsell, R. Lim, J. Beutel, and L. Thiele. The Design of a Responsive and Energy-Efficient Event-Triggered Wireless Sensing System. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, EWSN '17, pages 144–155. Junction Publishing, 2017.
- [SDL09] G. Senfaute, A. Duperret, and J. A. Lawrence. Micro-Seismic Precursory Cracks Prior to Rock-Fall on Coastal Chalk Cliffs: A Case Study at Mesnil-Val, Normandie, NW France. *Natural Hazards and Earth System Sciences*, 9:1625–1641, October 2009.

- [SGZ20] A. Saeed, D. Grangier, and N. Zeghidour. Contrastive Learning of General-Purpose Audio Representations. *arXiv:2010.10915 [cs, eess]*, October 2020.
- [SHK⁺14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, January 2014.
- [SHW⁺10] F. Stajano, N. Hault, I. Wassell, P. Bennett, C. Middleton, and K. Soga. Smart bridges, smart tunnels: Transforming wireless sensor networks from research prototypes into robust engineering infrastructure. *Ad Hoc Networks*, 8(8):872–888, November 2010.
- [SJS⁺21] D. H. Shugar, M. Jacquemart, D. Shean, S. Bhushan, K. Upadhyay, A. Sattar, W. Schwanghart, S. McBride, M. V. W. de Vries, M. Mergili, A. Emmer, C. Deschamps-Berger, M. McDonnell, R. Bhambri, S. Allen, E. Berthier, J. L. Carrivick, J. J. Clague, M. Dokukin, S. A. Dunning, H. Frey, S. Gascoin, U. K. Haritashya, C. Huggel, A. Käb, J. S. Kargel, J. L. Kavanaugh, P. Lacroix, D. Petley, S. Rupper, M. F. Azam, S. J. Cook, A. P. Dimri, M. Eriksson, D. Farinotti, J. Fiddes, K. R. Gnyawali, S. Harrison, M. Jha, M. Koppes, A. Kumar, S. Leinss, U. Majeed, S. Mal, A. Muhuri, J. Noetzli, F. Paul, I. Rashid, K. Sain, J. Steiner, F. Ugalde, C. S. Watson, and M. J. Westoby. A massive rock and ice avalanche caused the 2021 disaster at Chamoli, Indian Himalaya. *Science*, July 2021.
- [SKH⁺21] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo. “Everyone wants to do the model work, not the data work”: Data Cascades in High-Stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, Yokohama Japan, May 2021. ACM.
- [SKP15] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, June 2015.
- [SMG⁺07] T. Spillmann, H. Maurer, A. G. Green, B. Heincke, H. Willenberg, and S. Husen. Microseismic Investigation of an Unstable Mountain Slope in the Swiss Alps. *Journal of Geophysical Research*, 112(B7), July 2007.
- [SMS15] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised Learning of Video Representations using LSTMs. In *ICML*, pages 843–852, 2015.
- [Sou] Sounds of Matterhorn. <https://soundcloud.com/user-921998996/>.
- [SPI08] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 614–622, New York, NY, USA, August 2008. Association for Computing Machinery.
- [Ste13] S. Steinarsson. *Downsampling Time Series for Visual Representation*. Thesis, June 2013.
- [STM16] STMicroelectronics. *STM32F765xx STM32F767xx STM32F768Ax STM32F769xx*, May 2016.
- [SWS⁺15] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals. Learning the Speech Front-End With Raw Waveform CLDNNs. In *Proc. Interspeech*, 2015.
- [SZ14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

- [SZDF⁺15] F. Sutton, M. Zimmerling, R. Da Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, and L. Thiele. Bolt: A Stateful Processor Interconnect. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, pages 267–280. ACM, 2015.
- [TGPG16] N. Takahashi, M. Gygli, B. Pfister, and L. V. Gool. Deep Convolutional Neural Networks and Data Augmentation for Acoustic Event Recognition. In *Proc. Interspeech 2016*, San Francisco, 2016.
- [THGT07] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin. PermaSense: Investigating permafrost with a WSN in the Swiss Alps. In *Proceedings of the 4th Workshop on Embedded Networked Sensors*, EmNets '07, pages 8–12, New York, NY, USA, June 2007. Association for Computing Machinery.
- [TKI19] Y. Tian, D. Krishnan, and P. Isola. Contrastive Multiview Coding. *arXiv:1906.05849 [cs]*, June 2019.
- [TKW20] H. Y. Teh, A. W. Kempa-Liehr, and K. I.-K. Wang. Sensor data quality: A systematic review. *Journal of Big Data*, 7(1):1–49, December 2020.
- [TMJS20] J. Tack, S. Mo, J. Jeong, and J. Shin. CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances. *arXiv:2007.08176 [cs, stat]*, October 2020.
- [TMS⁺20] M. Tkachenko, M. Malyuk, N. Shevchenko, A. Holmanyuk, and N. Liubimov. Label Studio: Data labeling software, 2020.
- [TSE⁺20] D. Tsipras, S. Santurkar, L. Engstrom, A. Ilyas, and A. Madry. From ImageNet to Image Classification: Contextualizing Progress on Benchmarks. *arXiv:2005.11295 [cs, stat]*, May 2020.
- [VBM18] T. D. Verykios, D. Balsamo, and G. V. Merrett. Selective Policies for Efficient State Retention in Transiently-Powered Embedded Systems: Exploiting Properties of NVM Technologies. *Sustainable Computing: Informatics and Systems*, July 2018.
- [VDOM19] M.-D. Van Damme, A.-M. Olteanu-Raimond, and Y. Méneroux. Potential of crowdsourced data for integrating landmarks and routes for rescue in mountain areas. *International Journal of Cartography*, 5(2-3):195–213, May 2019.
- [vH08] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
- [vS11] A. van Herwijnen and J. Schweizer. Monitoring avalanche activity using a seismic sensor. *Cold Regions Science and Technology*, 69(2):165–176, December 2011.
- [WAK⁺20] F. Walter, F. Amann, A. Kos, R. Kenner, M. Phillips, A. de Preux, M. Huss, C. Tognacca, J. Clinton, T. Diehl, and Y. Bonanomi. Direct observations of a three million cubic meter rock-slope collapse with almost immediate initiation of ensuing debris flows. *Geomorphology*, 351:106933, February 2020.
- [WAY⁺98] M. Withers, R. Aster, C. Young, J. Beiriger, M. Harris, S. Moore, and J. Trujillo. A comparison of select trigger algorithms for automated global seismic phase and event detection. *Bulletin of the Seismological Society of America*, 88(1):95–106, February 1998.
- [WBF⁺19] S. Weber, J. Beutel, R. D. Forno, A. Geiger, S. Gruber, T. Gsell, A. Hasler, M. Keller, R. Lim, P. Limpach, M. Meyer, I. Talzi, L. Thiele, C. Tschudin, A. Vieli, D. Vonder Mühl, and M. Yücel. A decade of detailed observations (2008–2018) in steep bedrock permafrost at the Matterhorn Hörnligrat (Zermatt, CH). *Earth System Science Data*, 11(3):1203–1237, August 2019.

- [WBR⁺20] J. Winkens, R. Bunel, A. G. Roy, R. Stanforth, V. Natarajan, J. R. Ledsam, P. MacWilliams, P. Kohli, A. Karthikesalingam, S. Kohl, T. Cemgil, S. M. A. Eslami, and O. Ronneberger. Contrastive Training for Improved Out-of-Distribution Detection. *arXiv:2007.05566 [cs, stat]*, July 2020.
- [WDF08] F. Walter, N. Deichmann, and M. Funk. Basal icequakes during changing subglacial water pressures beneath Gornergletscher, Switzerland. *Journal of Glaciology*, 54(186):511–521, 2008.
- [Web11] WebGL - OpenGL ES for the Web. <https://www.khronos.org/>, July 2011.
- [WFB⁺18] S. Weber, D. Fäh, J. Beutel, J. Faillettaz, S. Gruber, and A. Vieli. Ambient seismic vibrations in steep bedrock permafrost used to infer variations of ice-fill in fractures. *Earth and Planetary Science Letters*, 501:119–127, November 2018.
- [WFM⁺18] S. Weber, J. Faillettaz, M. Meyer, J. Beutel, and A. Vieli. Acoustic and Microseismic Characterization in Steep Bedrock Permafrost on Matterhorn (CH). *Journal of Geophysical Research: Earth Surface*, 123(6):1363–1385, May 2018.
- [WHv⁺21] M. Wenner, C. Hibert, A. van Herwijnen, L. Meier, and F. Walter. Near-real-time automated classification of seismic signals of slope failures with continuous random forests. *Natural Hazards and Earth System Sciences*, 21(1):339–361, January 2021.
- [WJR⁺05] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring Volcanic Eruptions with a Wireless Sensor Network. In *Proceedings of the Second European Workshop on Wireless Sensor Networks, 2005.*, pages 108–120, February 2005.
- [WJWH⁺18] K. Wikstrom Jones, G. J. Wolken, D. Hill, R. L. Crumley, A. A. Arendt, J. Joughin, and L. Setiawan. Community Snow Observations (CSO): A Citizen Science Campaign to Validate Snow Remote Sensing Products and Hydrological Models. 2018:IN22B–01, December 2018.
- [WLJ⁺06] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation, OSDI '06*, pages 381–396. USENIX Association, 2006.
- [WLR⁺06] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, March 2006.
- [WLW⁺16] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized Convolutional Neural Networks for Mobile Devices. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4820–4828, June 2016.
- [XCW⁺15] S. H. I. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015.
- [XGF16] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning (ICML)*, 2016.
- [YLW⁺18] S. Yuan, J. Liu, S. Wang, T. Wang, and P. Shi. Seismic Waveform Classification and First-Break Picking Using Convolution Neural Networks. *IEEE Geoscience and Remote Sensing Letters*, 15(2):272–276, February 2018.

-
- [YOBB15] C. E. Yoon, O. O'Reilly, K. J. Bergen, and G. C. Beroza. Earthquake detection through computationally efficient similarity search. *Science Advances*, 1(11), 2015.
- [Zar21] Zarr. Zarr Developers, September 2021.
- [ZCJ16] Y. Zhang, W. Chan, and N. Jaitly. Very Deep Convolutional Networks for End-to-End Speech Recognition. *arXiv: 1610.03022*, October 2016.
- [ZSLC17] Y. Zhang, N. Suda, L. Lai, and V. Chandra. Hello Edge: Keyword Spotting on Microcontrollers. *arXiv:1711.07128 [cs, eess]*, November 2017.
- [ZSLM04] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi. Hardware Design Experiences in ZebraNet. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04*, pages 227–238, New York, NY, USA, 2004. ACM.
- [ZSS11] L. Zhao, G. Sukthankar, and R. Sukthankar. Incremental Relabeling for Active Learning with Noisy Crowdsourced Annotations. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 728–733, October 2011.
- [ZYG+17] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights. *arXiv:1702.03044 [cs]*, February 2017.
- [ZZL+08] X. Zhou, X. Zhuang, M. Liu, H. Tang, M. Hasegawa-Johnson, and T. Huang. HMM-Based Acoustic Event Detection with AdaBoost Feature Selection. In R. Stiefelhagen, R. Bowers, and J. Fiscus, editors, *Multimodal Technologies for Perception of Humans*, number 4625 in Lecture Notes in Computer Science, pages 345–353. Springer Berlin Heidelberg, 2008.

Credits

Cover image taken from flickr.com by simonsimages. It is licensed under the Creative Commons Attribution 2.0 Generic license.
<https://www.flickr.com/photos/simonsimages/>

