Machine-Learning-based Background Identification for the Radio Neutrino Observatory Greenland

Master's Thesis in Physics

Presented by

Philipp Laub

04.12.2023

Erlangen Centre for Astroparticle Physics
Friedrich-Alexander-Universität Erlangen-Nürnberg

Supervisor: Prof. Dr. Anna Nelles

Abstract

The Radio Neutrino Observatory Greenland (RNO-G), an in-ice radio neutrino detector, is currently being built to detect ultra-high energy (UHE) neutrinos above 10 PeV by measuring radio waves, which are created via the Askaryan effect when UHE neutrinos interact in the ice. Located at Summit Station in Greenland, RNO-G is designed as a wide-spread station array, where each station is equipped with several radio antennas, which are positioned either close to the surface or deep in the ice.

Despite the remote and relatively radio-quiet location at Summit, various backgrounds, often of anthropogenic origin, occur in the radio frequency regime. Furthermore, there is a rather unexplored background that appears only during time periods of high wind speed. These "wind events" are impulsive signals of unknown origin, appear in different forms, and can in some stations make up the majority of impulsive signals triggered by the shallow part of a station during high-wind periods. Since only little is known about this wind-related background and other backgrounds are present in the data, it is difficult to analyze wind events separately from other events. Therefore, an artificial neural network, namely a variational autoencoder (VAE), is used in this thesis to represent measured events in a low-dimensional space of features. Events are then clustered in this space using the clustering algorithm HDBSCAN. The obtained clusters are analyzed, primarily with respect to wind speed, to extract wind event clusters from the bulk of available data. Various wind clusters are found for three stations, where the number of clusters and their characteristics strongly vary among different stations. Furthermore, a still unknown background class, which occurs periodically and only during daytime, is found for one station.

During high wind speed periods, wind events can pose a considerable background and might interfere with analyses. Hence, another VAE is trained on previously selected wind events of a single station to discriminate between wind events and non-wind events via anomaly detection. For the station the VAE is trained on, the method allows the separation of wind events from non-wind events, including other backgrounds and cosmic ray simulations. It is for example possible to reject 95 % of test wind data while less than 1 % of cosmic ray simulations are falsely classified as wind. However, an evaluation of other stations suggests that a VAE trained on one station is not usable for other stations, since wind events appear to vary across stations.

Contents

1. Introduction

Of all the elementary particles of the Standard Model neutrinos are among the most elusive ones. Appearing in three flavors, neutrinos interact only via the weak interaction and gravity. Their masses are not yet known but are so small that only upper limits were experimentally determined [1], making the impact of gravity for a single neutrino negligible. Furthermore, neutrinos do not carry electric charge and are thus not deflected by magnetic fields. Since neutrinos do not interact via the strong and electromagnetic forces, they interact only very rarely with matter. But it is precisely these properties that make neutrinos very important particles for multi-messenger astronomy since the arrival direction of neutrinos points straight back to their sources.

Over the past decades, multiple neutrino observatories have been built, including IceCube [2] and KM3NeT [3]. Since neutrinos interact so rarely with matter, huge volumes or a large neutrino flux are required to detect neutrinos. Hence, either water (KM3NeT) or ice (IceCube) is usually chosen as a detection medium, since both are abundant and inexpensive. When high-energy neutrinos interact with ice (water), charged leptons are created which can emit Cherenkov radiation that is then captured by the detector. However, this technique requires a rather dense array of detection modules. For ultra-high energy (UHE) neutrinos, the predicted flux is even smaller, and therefore large volumes must be covered by the detector to be able to detect UHE neutrinos within a reasonable amount of time.

This is where radio detectors such as the Radio Neutrino Observatory Greenland (RNO-G) [4], which is aimed to detect neutrinos via Askaryan radiation in ice, start to shine since the attenuation length of radio waves in ice is up to $\sim 1\,\mathrm{km}$ [5]. Therefore, a single radio detection station by itself can cover a huge volume, allowing a sparse but in return wide-spread station array to cover immense volumes of ice.

However, in the radio frequency regime, a lot of background noise is present, including man-made backgrounds emitted by e.g. hand-held radio devices but also signals from e.g. the station's wireless communication system [6]. Furthermore, not all backgrounds are anthropogenic: it has been observed that the event rate increases with increasing wind speed [7]. In [7] the authors investigate the possi-

bility that these "wind events" are linked to the triboelectric effect and that radio signals are emitted from coronal discharge due to electrostatic potentials forming between air and ice/snow layers. However, so far no explicit measurements that try to simulate the conditions using e.g. a wind tunnel and snow were performed. Still, these events pose a considerable temporary background up to the point where wind-correlated events make up the majority of impulsive shallow-triggered events that triggered various RNO-G stations during high-wind periods.

The aim of this thesis is to investigate and learn more about these seemingly wind-induced events for RNO-G and find a method (here using artificial neural networks and machine learning algorithms) to discriminate them from other events so that they do not interfere with other analyses.

Therefore, RNO-G is introduced in chapter 2 and known background events (especially wind-induced events) are presented in more detail in chapter 3. Artificial neural networks and machine learning methods are introduced in chapter 4. Chapter 5 describes how wind events were identified and chapter 6 presents the method chosen to filter wind events. Finally, a summary as well as conclusions and an outlook can be found in chapter 7.

2. The Radio Neutrino Observatory Greenland

In this chapter, the Radio Neutrino Observatory Greenland (RNO-G) is presented. RNO-G is an in-ice neutrino detector that is currently being built to detect UHE neutrinos above 10 PeV energies using the in-ice radio detection technique described in the following [4].

2.1. Radio detection of neutrinos

If a UHE neutrino interacts with particles in the ice via deep inelastic scattering, a shower of secondary particles is created. This is shown in Figure 2.1. The incoming neutrino propagates through the ice and undergoes a weak interaction process at the vertex. The created secondary particles (represented by yellow circles in Figure 2.1) propagate through the ice and create radio emission primarily due to the *Askaryan effect* [8]: The positrons created in the shower annihilate with electrons of the surrounding medium. Additionally, the medium is further ionized due to scattering processes such as Compton scattering. This results in an immobile positive net charge (ionized medium) and a moving negative net charge (shower front) leading to the emission of electromagnetic radiation in the radio frequency regime (from ∼MHz up to ∼GHz). The radiation is coherent near the Cherenkov cone (shown in blue in Figure 2.1) and strongly polarized. Coherence is lost when going away from the cone, where the high frequencies are affected first [4]. This is shown in Figure 2.2 for a radio signal originating from a hadronic shower, where the power in the higher frequencies decreases with increasing angle away from the Cherenkov cone. Apart from the Askaryan effect, *geomagnetic emission* can create radio emission as well: Charged particles are deflected by Earth's magnetic field due to the Lorentz force, resulting in radio emission due to charge separation, since particles with opposite charge are deflected in opposite directions. In ice, however, the radio emission is dominated by the Askaryan effect [4]. A radio signal detected at a detector station (composed of a shallow and a deep part) has either propagated on a straight or bent path, depending on the varying index of refraction of ice.

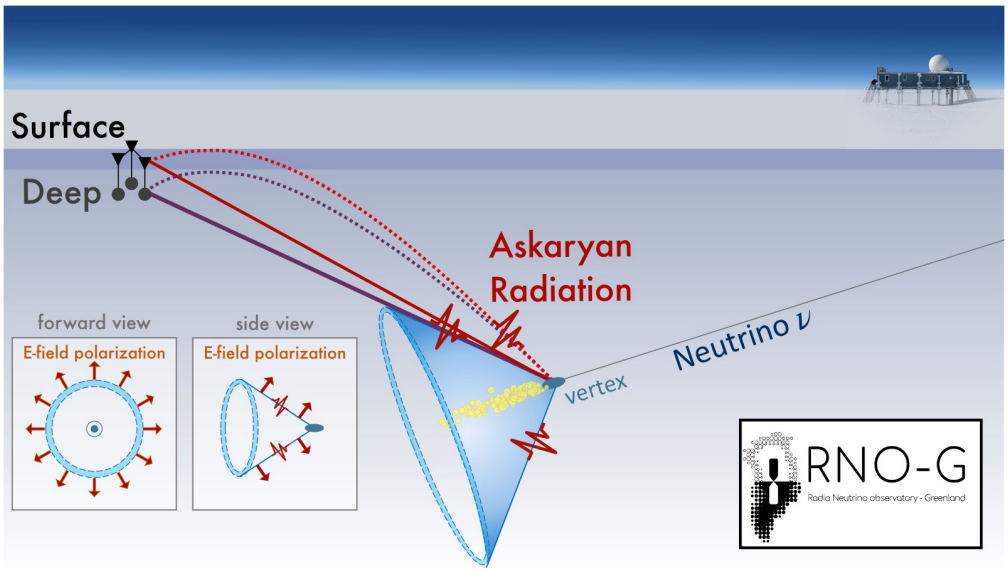


Figure 2.1.: Sketch of a neutrino interaction and subsequent radio emission due to the Askaryan effect in ice. The polarized radio signal propagates on straight or bent trajectories to the detector (here composed of deep and surface antennas). Figure taken from [4].

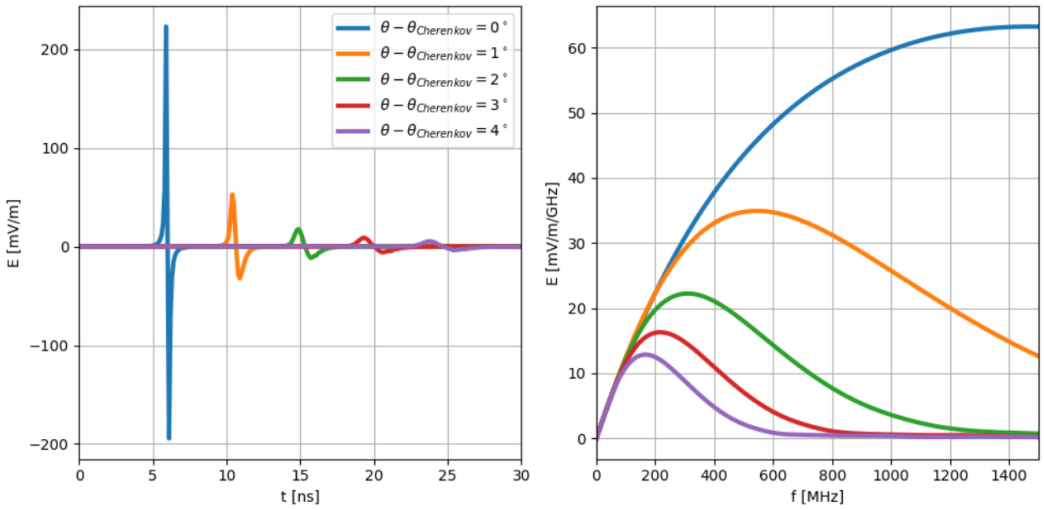


Figure 2.2.: Electric-field waveforms (left) and spectra (right) of a radio signal created by a hadronic shower with deposited energy of 1 EeV. Shown are signals for different viewing angles relative to the Cherenkov angle. The waveforms have been offset in time for better readability. Propagation effects and detector responses have not been included. Figure taken from [4].

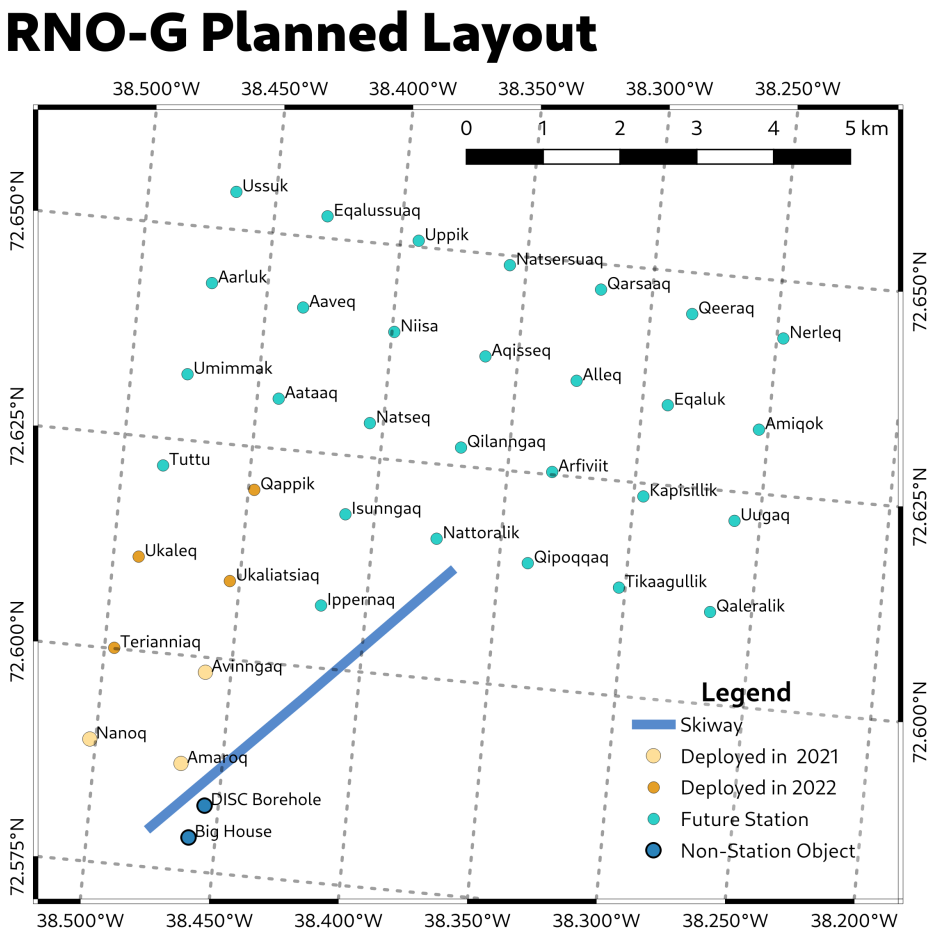


Figure 2.3.: Map of the planned RNO-G station layout. Seven of 35 stations are already built, where the color encodes the year of deployment. The figure shown here is cropped and the original figure is created by the RNO-G collaboration.

2.2. Layout of the RNO-G detector

RNO-G is located at Summit Station (Greenland), on top of the ice sheet at a height of more than 3 km. The location is fairly isolated from human civilization so external man-made backgrounds are reduced to a minimum. Figure 2.3 shows the planned RNO-G detector layout. Of the 35 planned stations, 7 have already been deployed in 2021 and 2022, namely stations 11 (Nanoq, 2021), 12 (Terianniaq, 2022), 13 (Ukaleq, 2022), 21 (Amaroq, 2021), 22 (Avinngaq, 2021), 23 (Ukaliatsiaq, 2022) and 24 (Qappik, 2022), where the spacing between neighboring stations is 1.25 km. The attenuation length of radio signals in ice is up to ~ 1 km [5].

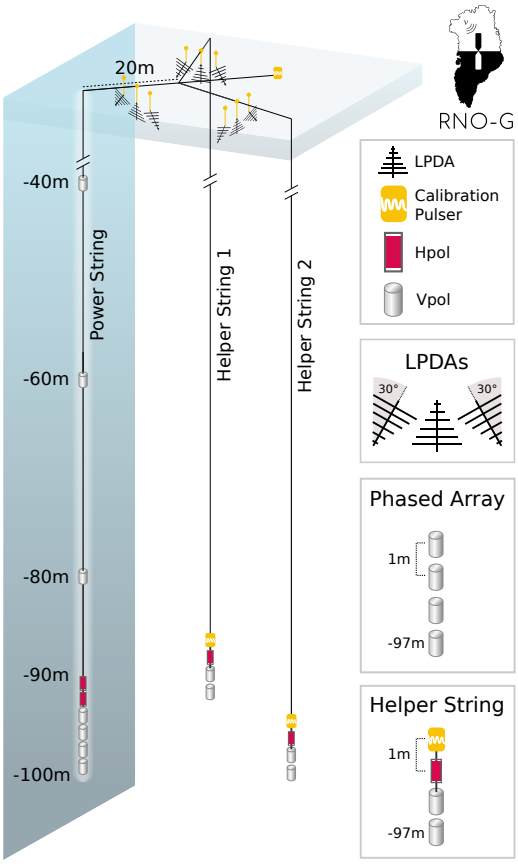


Figure 2.4.: Schematic of the layout of an RNO-G station. The shallow part consists of a calibration pulser and nine log-periodic dipole antennas (LPDAs) arranged in groups of three and oriented as depicted in the legend. The deep part consists of horizontally and vertically polarized antennas (Hpol and Vpol) distributed across three antenna strings (power and two helper strings) up to 100 m deep in the ice. Additionally, each of the helper strings is equipped with a calibration pulser. Figure created by the RNO-G collaboration.

The layout of an RNO-G station is sketched in Figure 2.4. It consists of a shallow and a deep part. The deep part is composed of three antenna strings that carry horizontally and vertically polarized antennas (Hpol and Vpol) at various depths as well as calibration pulsers. The power string features the *phased array* that is able to determine the inclination of an arriving signal, while the azimuth is determined using the helper strings. The shallow part consists of nine log-periodic dipole antennas (LPDAs) arranged in groups of three antennas per string buried in trenches about 1.5 m deep in the ice/firn. Per group, the antennas are oriented in such a way that the one in the center is pointing upwards and the outer ones are

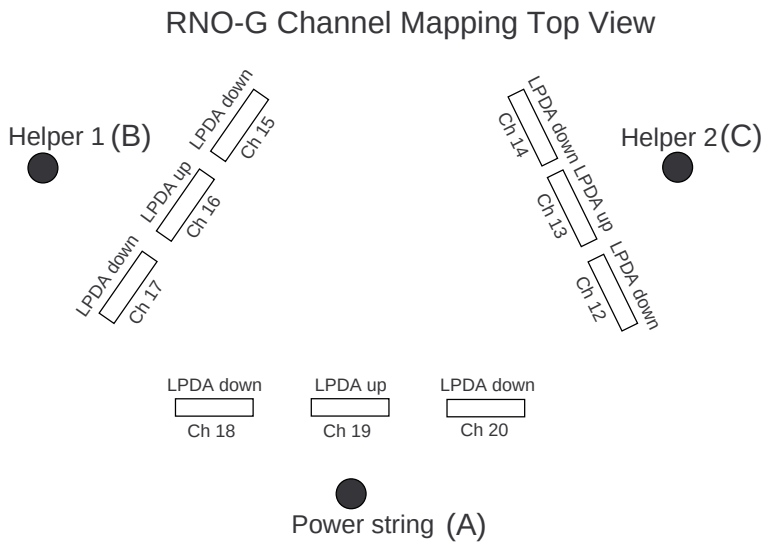


Figure 2.5.: Top view of the shallow part of an RNO-G station. The LPDAs (rectangles) are arranged in sets of three with two downward-facing antennas and one upward-facing antenna per set. Per group, the antennas are placed in a line roughly perpendicular to the connection between its nearest antenna string and the center of the triangle spanned by all antenna string points. Figure created by the RNO-G collaboration.

pointing downwards with an inclination of 30 °. A top-down view of the shallow part is shown in Figure 2.5. The antennas of each of the three sets are positioned in a line that is roughly perpendicular to the connection between their nearest antenna string point and the center of the triangle spanned by the three string points. The channel numbers assigned to each LPDA are denoted in Figure 2.5 as well, whereby the channel numbers of the upward-facing antennas are 13, 16, and 19. In terms of power management, stations are equipped with solar panels and/or wind turbines, where the wind turbines are required to increase active observation time during winter.

2.3. Cosmic rays

The upward-facing shallow antennas are not meant to measure neutrinos. This is the purpose of the deep antennas and downward-facing LPDAs since the interaction point of a neutrino is expected to lie somewhere in the ice and the sensitivity of the LPDAs is low at the backside. Instead, the upward-facing antennas act as a veto for signals coming from above, which are not created by UHE neutrinos. Among the sources of these signals are cosmic rays, high-energy charged particles,

which originate from outside or within our galaxy. When cosmic rays interact with Earth's atmosphere, extensive air showers of secondary particles are created. In contrast to ice, the radio emission for cosmic rays in air or rather their secondary particles is dominated by the geomagnetic effect [4]. Therefore, the upward-facing antennas can be used to indirectly detect and study cosmic rays in the radio regime. The expected rate at which cosmic rays are detected strongly depends on the trigger and is estimated to be ~ 1 cosmic ray per day per station [9].

2.4. Triggers and saved data

The signal recorded with an antenna – the waveform (also called trace) – is an amplitude (corresponding to voltage) as a function of time. Since the antenna output is not saved continuously, a trigger system is required to initiate the saving process for a signal for all antennas. Per station, there are multiple triggers active with the most important ones being the force trigger, the deep trigger, and the radiant (surface/shallow) trigger:

- The *force trigger* triggers a recording at a constant frequency of 0.1 Hz regardless of whether there is a signal or not, and therefore most events triggered by the force trigger are thermal noise.
- The *radiant (surface/shallow) trigger* triggers if the approximated Hilbert envelope of a signal exceeds a certain threshold within a coincidence window of 50 ns in either two out of six downward-facing LPDAs or two out of three upward-facing LPDAs. Hereby the trigger rate is tried to be kept constant, so the trigger thresholds are adjusted per channel via a PID[1] controller to meet a certain target trigger rate, which is different across stations.
- The *deep trigger* is a two out of four channels coincidence trigger of the four lowest antennas of the power string (the phased array) with a coincidence window of 30 ns.

Due to strongly varying cable lengths and resulting varying signal runtimes, the timing between shallow and deep channels is not ideal, so if the shallow trigger fires, the actual signal is not present in the recorded data of the deep antennas. This problem is planned to be solved in the near future.

At the moment, waveforms are recorded with a sampling rate of 3.2 GHz and have a length of 2048 samples or 640 ns. An example waveform and spectrum of

[1] A proportional–integral–derivative (PID) controller is a mechanism that uses a feedback control loop to approximate a measured variable/quantity to a chosen setpoint by adjusting controllable parameters based on the error between setpoint and measured variable [10].

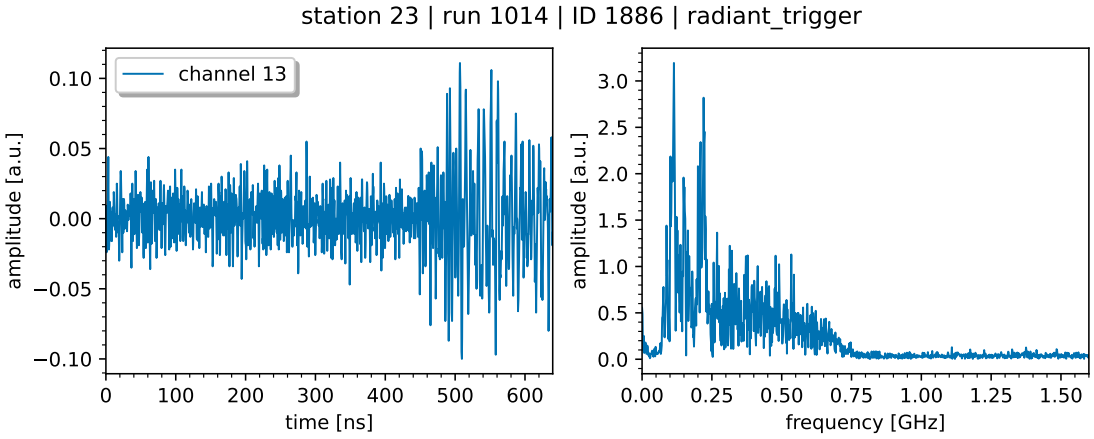


Figure 2.6.: Waveform and corresponding frequency spectrum of an event triggered by the radiant (shallow) trigger and captured by channel 13 of station 23. Note that the signal is cut off at the end.

an impulsive event that triggered the shallow (radiant) trigger and was recorded with channel 13 is shown in Figure 2.6. One can immediately see that the signal is cut off at the end. This is due to the position of the trigger resulting in the actual signal appearing only in the second half of the recorded waveform. Longer waveforms are therefore clipped at the end. To counter this problem, the sampling rate will be reduced to 2.4 GHz in the future, increasing the time interval of saved waveforms. Currently, the channels are not yet calibrated, so waveform amplitudes in this thesis are given in arbitrary units but are of the same scale.

The antenna signals are amplified between 80 MHz and 800 MHz and filtered outside of this band as can be seen in the spectrum. Furthermore, the LPDAs are more sensitive to lower frequencies around 100 MHz than to higher frequencies due to the LPDA's vector effective length, i.e. the conversion between E-field and voltage.

Data is split up into *runs*, where one full run equals two hours of data under normal conditions. The data is saved to disk at Summit Station but the full data can not be transmitted via satellite due to transfer rate limitations. Therefore, the full data is hand-carried to one of RNO-G's institutions after a deployment season to transfer them onto a server. Only a small fraction (e.g. 5 %) of the data, the *sub-sampled* data, is transferred directly via satellite. This data set is free to use for analyses, while the rest of the data is usually hidden (*blinded*) to prevent potential biases created from knowledge about the full data. Furthermore, trigger information of the full data set is stored in *header* files, which are transferred via

satellite as well.

In this thesis, the reconstruction framework NuRadioReco [11] is used to handle RNO-G data.

3. Backgrounds

As noted in chapter 2, the upward-facing shallow antennas are intended to measure and veto signals coming from above. Apart from cosmic rays, these signals are mainly backgrounds from non-astrophysical sources. The main backgrounds observed so far are discussed in this chapter with a special focus on wind-correlated events.

3.1. Thermal noise

The background that occurs in all antennas at any time is thermal noise. It is created by electrons in a conductor moving due to their thermal energy independent of the applied voltage. For thermal noise, each frequency is equally likely resulting on average in a flat frequency spectrum. Thermal noise is always present in recorded waveforms and can sometimes randomly exceed the chosen trigger threshold, triggering an event if the coincidence condition is satisfied. To study thermal noise, the force trigger periodically triggers an event, regardless of the signal amplitude. An example waveform and corresponding frequency spectrum of an event triggered by the force trigger is shown in Figure 3.1. This event is assumed to be dominantly thermal noise.

Apart from pure thermal noise, the upward-facing antennas measure contributions from galactic noise as well [12].

3.2. Anthropogenic backgrounds

In this part, some man-made background noises are presented, including battery charging, Long Range Wide Area Network (LoRaWAN), wind turbine noises, and continuous wave (CW) events.

3.2.1. Battery charging and LoRaWAN noise

For stations deployed in 2021, an increase in event rate was observed during daytime and battery charging from solar panels was identified as the source of these events (see example in Figure 3.2). Furthermore, impulsive events (example in Figure 3.3) occurring every minute were observed, which correspond to data-sending

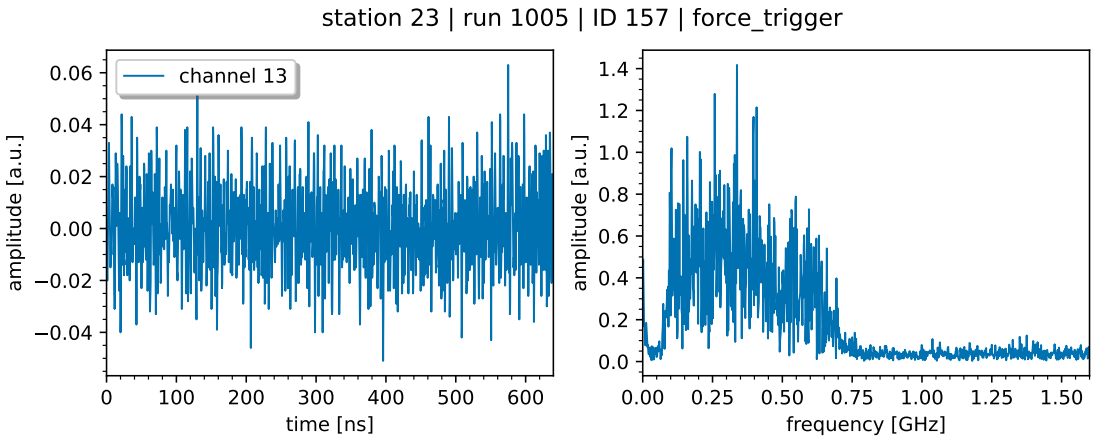


Figure 3.1.: Channel 13 waveform and frequency spectrum of an event triggered by the force trigger. The signal is assumed to be dominantly thermal noise.

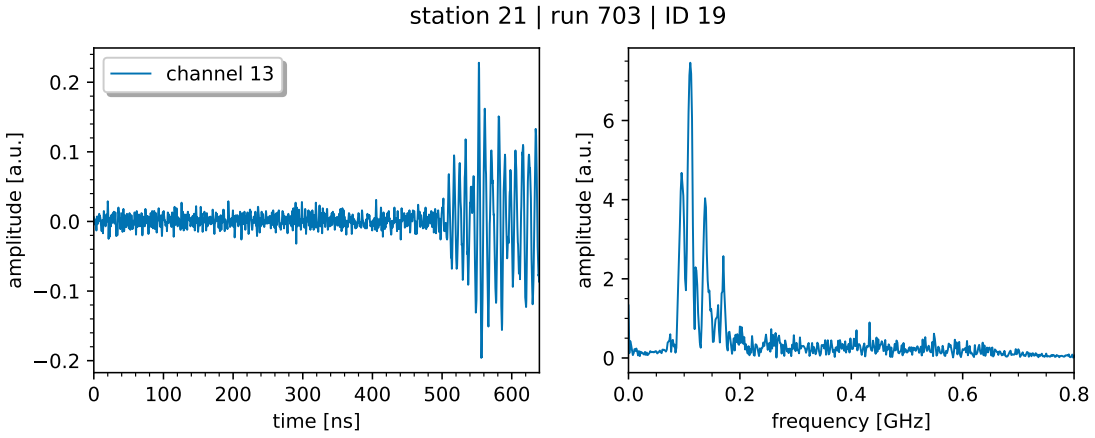


Figure 3.2.: Channel 13 waveform and frequency spectrum of an event attributed to battery charging.

intervals of the LoRaWAN communication system that transfers data from the detector station to the main station at Summit. Changes to the station hardware were made so that stations deployed in 2022 (12, 13, 23, 24) and station 11 should not show these backgrounds anymore.

3.2.2. Wind turbine noise

Stations 11 and 12 are equipped with wind turbines to increase active observation time during winter. However, these wind turbines caused increased shallow trigger rates (independent of wind speed) and so far all attempts to remove this

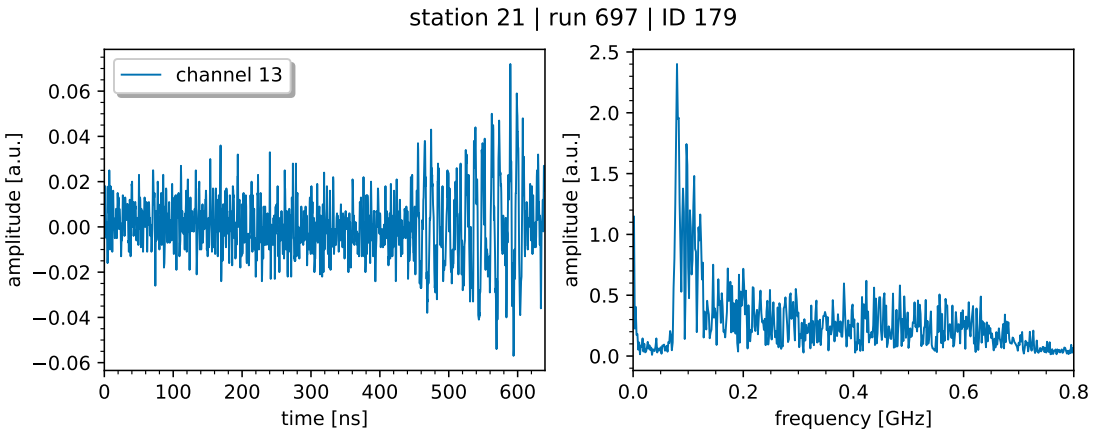


Figure 3.3.: Channel 13 waveform and frequency spectrum of an event attributed to LoRaWAN noise.

background failed. New attempts to eliminate wind turbine noise are planned for 2024.

3.2.3. Continuous wave events

Another class of anthropogenic backgrounds is continuous wave (CW) events, which exhibit a narrowband spike at a certain frequency in the spectrum (see example in Figure 3.4). Known sources are air traffic at $\sim$ 125 MHz, handheld radio devices (Walkie-Talkies) at $\sim$ 136 MHz and $\sim$ 152 MHz and the weather balloon at 403 MHz. Furthermore, additional CW lines at e.g. 200 MHz and 480 MHz are observed, some of which might be station-induced but their sources and origin remain unknown [12].

To identify CW events, one can use the *L1 score*, which was developed for ARIANNA [13]. For each frequency bin of a frequency spectrum, the fraction of the power in that bin compared to the total power of the whole spectrum is calculated. The L1 score is then the maximum over these ratios. If the L1 score exceeds a certain threshold (e.g. 0.6) in any of the channels, the event is classified as a CW event.

3.2.4. Other anthropogenic backgrounds

Apart from CW, battery charging, LoRaWAN, and wind turbine noise, there are further backgrounds originating from human activity, including airplanes, heavy machinery, and snowmobiles [6]. Snowmobiles are used e.g. as part of the IceSat-2 mission [14], where snow depth measurements are taken once a month.

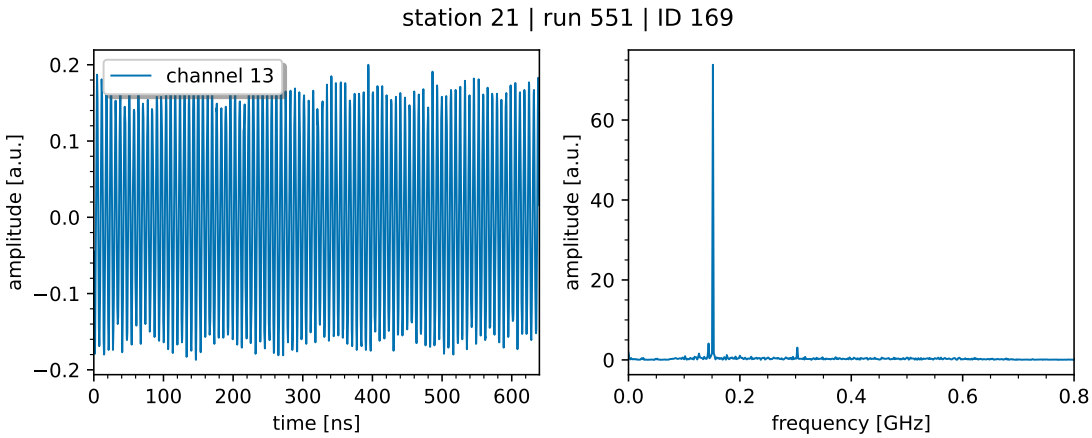


Figure 3.4.: Channel 13 waveform and frequency spectrum of an event identified as narrowband (CW) noise.

3.3. Physics backgrounds

Besides the anthropogenic backgrounds, cosmic ray air showers and solar flares can create radio emission as well [6]. Furthermore, radio signals stemming from cosmic-ray-induced atmospheric muons may look similar to neutrino events in ice [15]. Moreover, as mentioned before, galactic noise is detected in the upward-facing antennas.

3.4. Wind-correlated background

Lastly, backgrounds correlated with high wind speed are discussed. In [7], the authors report that all radio neutrino experiments noticed the appearance of signals in the radio regime when the wind velocity exceeded a certain threshold. This threshold varies from experiment to experiment, potentially dependent on antenna type and trigger threshold, but lies at around 10 m/s for most experiments with some experiments reporting lower thresholds of 6−8 m/s and some reporting higher thresholds up to 12 m/s. Triggered event rates were observed to increase with increasing wind speed after a turn-on, where the wind speed exceeded a threshold. Furthermore, some experiments found that wind-correlated signals were stronger for higher wind velocities [7].

As described in [7], a possible explanation for the occurrence of these wind-correlated events is a displacement of charges by wind blowing over snow via the triboelectric effect. This can create an electric field and can lead to the emission of electromagnetic radiation after a coronal discharge. Accumulation of charges on

blocks of ice and development of electrostatic potentials due to the wind blowing over ice/snow was indeed measured by various experiments, which report wind speed thresholds similar to those described above [16–19]. Using laboratory tests, the authors of [7] demonstrated that a coronal discharge (simulated by spark discharge) can emit a signal in the radio regime with a time duration of the order of 10 ns, corresponding to the expected timescale of a neutrino or cosmic ray signal measured by radio detectors.

For some radio neutrino detectors, it was possible to reconstruct the source location of wind-correlated events to be mostly at or near man-made structures, which are (partially) made of metallic material. Interestingly, similar observations were made in aviation, where airplanes flying through precipitation experienced radio noise that interfered with communication systems. The noise was attributed to frictional charging by precipitation particles and subsequent discharges [20]. However, apart from discharges on metallic surfaces, it is still possible that events originating from the triboelectric effect are produced somewhere on the snow/ice. In this case, a homogeneous distribution of the source location would be expected.

Independent of how these wind-correlated events are created, they pose a significant temporary background and may interfere with analyses that use the upward-facing antennas e.g. cosmic ray analyses. For RNO-G at Summit Station (Greenland), the fraction of time the wind speed was $\geq 10\,\mathrm{m/s}$ in 2021 and 2022 is 11 %. If a more pessimistic turn-on of 7 m/s is assumed, this fraction increases to 35 %. Hence, it is crucial to further investigate this phenomenon and find a way to reject this background without rejecting the total high-wind time period.

4. Machine learning

In this thesis, *artificial neural networks* (ANNs) and machine learning algorithms are used to identify and discriminate wind events. At first, the structure and training of ANNs are presented in general. Additionally, two machine learning algorithms, UMAP and HDBSCAN, are introduced. After that, autoencoders and variational autoencoders are discussed. Parts of this chapter (sections 4.1 to 4.6 and 4.9) follow the lecture by Florian Marquardt [21].

4.1. Structure of artificial neural networks

An ANN is a (typically non-linear) function $F_\theta(x_{in}) = y_{out}$ with parameters $\theta = (\theta_1, \theta_2, \theta_3, ...)$ that maps an input x_{in} to an output y_{out} and aims to approximate a target output y_{target}. On a more fundamental level, ANNs are composed of *layers* of *neurons* (nodes) that have a value (usually a real number) and are connected to other neurons via *weights*.

Figure 4.1 shows the structure of a basic ANN. From left to right there is first the *input layer* consisting in this case of three neurons (depicted as circles). The input layer is connected to the next layer via weights (here depicted as lines). The second layer is then connected to the third layer until the penultimate layer is connected to the *output layer* on the right. Layers that are neither input nor output layers are called *hidden layers*. In this example, all neurons of one layer are connected to all neurons in the next layer. Therefore, these layers are called *fully-connected* (FC) layers. ANNs that consist of fully-connected input, output, and at least one hidden layer are called *multilayer perceptrons* (MLPs). If, as is the case in this example, the information in an ANN flows exclusively in one direction, i.e. from the input layer through hidden layers to the output layer (from left to right), such an ANN is called *feedforward neural network*.

The flow of information between two fully connected layers n and $n+1$ is given as follows: The value of neuron i of a layer n is denoted $y_i^{(n)}$. The weight that connects neuron $y_i^{(n)}$ to neuron $y_j^{(n+1)}$ is denoted $w_{ji}^{(n+1,n)}$. Additionally, a *bias* may be assigned to each neuron, where the bias of neuron $y_j^{(n+1)}$ of layer $n+1$ is called $b_j^{(n+1)}$. Together, weights and biases compose the parameters θ of the ANN. To

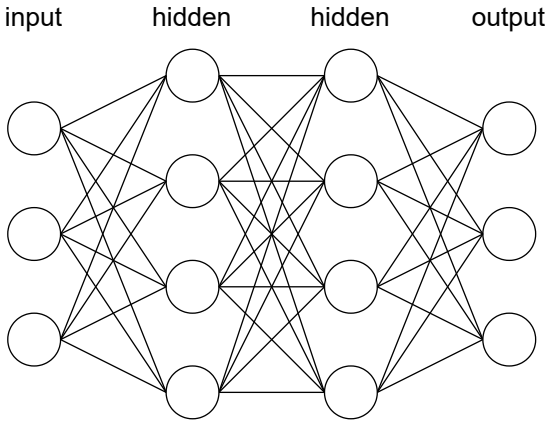


Figure 4.1.: Structure of an ANN (here: multilayer perceptron) with input, output, and two hidden layers. Neurons are depicted as circles and weights as lines.

compute the value of neuron $y_j^{(n+1)}$ of layer $n+1$ the quantity $z_j^{(n+1)}$ is calculated:

$$z_j^{(n+1)} = \sum_{i=1} y_i^{(n)} \cdot w_{ji}^{(n+1,n)} + b_j^{(n+1)} \tag{4.1}$$

This can also be written in terms of vectors and matrices:

$$\mathbf{z}^{(n+1)} = W^{(n+1,n)} \cdot \mathbf{y}^{(n)} + \mathbf{b}^{(n+1)} \tag{4.2}$$

$y_j^{(n+1)}$ is then given as

$$y_j^{(n+1)} = a_{n+1}(z_j^{(n+1)}), \tag{4.3}$$

where $a_{n+1}(z)$ is called the *activation function* of layer $n+1$. The activation function is typically non-linear, but sometimes the (linear) identity function $a_{n+1}(z_j^{(n+1)}) = z_j^{(n+1)}$ is chosen. Commonly used non-linear activation functions are the *rectified linear unit* (ReLU) and its variants and the *standard logistic function* (often referred to as *sigmoid*):

$$\mathrm{ReLU}(x) = \max(0, x) \tag{4.4}$$

$$\mathrm{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{4.5}$$

4.2. Parameter adjustment for artificial neural networks

The goal of an ANN F is to take an input and produce an output close to a target output i.e. the ANN aims to approximate a target function F_{target}:

$$F_\theta(x_{in}) = y_{out} \overset{!}{\approx} y_{target} = F_{target}(x_{in}) \tag{4.6}$$

Here x_{in} (y_{out}) are vectors, where each component $[x_{in}]_g$ ($[y_{out}]_g$) represents the value of a neuron of the input (output) layer.

To achieve a close approximation of the target function, one can repeatedly adjust the parameters θ of the neural network and compare the produced output to the target output. This process is called *training*. To compare y_{out} and y_{target}, a *cost function* $C(x_{in}, \theta)$ (also referred to as *loss function*) is chosen. Common choices for floating-point outputs are the *mean squared error* (MSE) and the *mean absolute error* (MAE):

$$\mathrm{MSE}(x_{in}) = \left\langle (F_\theta(x_{in}) - F_{target}(x_{in}))^2 \right\rangle \tag{4.7}$$

$$\mathrm{MAE}(x_{in}) = \left\langle \, | \, F_\theta(x_{in}) - F_{target}(x_{in}) \, | \, \right\rangle \tag{4.8}$$

Typically, the cost function is defined such that it is minimal if the neural network output is closest to the target output. Therefore, the goal is to minimize the cost function by adjusting the trainable parameters θ_s according to

$$\theta_s \to \theta_s + \delta\theta_s \tag{4.9}$$

using an *optimizer* such as *gradient descent*. Using gradient descent, the update of parameter θ_s is given by

$$\delta\theta_s = -\eta \frac{\partial C(x_{in}, \theta)}{\partial \theta_s}, \tag{4.10}$$

where η is the so-called *learning rate*. As a real-world analogy, one can think of this as moving down a hill or potential, where the learning rate determines the size of the steps. Choosing a suitable η is crucial because if it is too small, training might be slow and one may be trapped in a local minimum. This is because, coming back to the analogy of the potential, the energy required to overcome a local maximum to move from a higher to a lower local minimum might not be high enough. If, on the other hand, the learning rate is too large, one may overshoot and not converge to a minimum. Furthermore, very large learning rates are problematic in combination with activation functions $a(z)$ whose derivative converges to zero for $z \to \infty$ and/or $z \to -\infty$. If the update step is too large and the network

falls into one of these vanishing-derivative regions, the gradient and therefore the parameter update is vanishingly small and the network is then trapped in this state as it has no chance of leaving the zero-gradient zone. This problem is known as the *vanishing gradient problem* and occurs e.g. for the ReLU activation function. To prevent the gradient from vanishing, the ReLU is, if possible, modified into a LeakyReLU:

$$\text{LeakyReLU}(x;\alpha) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases} \tag{4.11}$$

α can be adjusted and as long as $\alpha \neq 1$, the LeakyReLU still remains non-linear.

Usually, a large number of inputs and target outputs is available for training. However, the cost function is usually defined to average over all possible training samples, so a single update step can become computationally very expensive. This problem is solved by randomly dividing the training data into *batches* of N_{batch} (the *batch size*) training samples. A training step is then performed on each batch rather than all samples at the same time. To update the trainable parameters, one averages over a batch:

$$\delta\theta_s = -\eta \left\langle \frac{\partial C(x_{in},\theta)}{\partial \theta_s} \right\rangle_{x_{in} \in batch} \tag{4.12}$$

This method is called *stochastic gradient descent*. Apart from stochastic gradient descent, other so-called *optimizers* are available such as the AdaGrad, RMSProp, Adadelta, and Adam optimizers [22]. The Adam [23] optimizer is based on the stochastic gradient descent, but includes a couple of modifications. Adam calculates exponential moving averages of both the gradient and the squared of the gradient, where the decay is controlled by some chosen parameters [23]. These moving averages estimate the mean and variance of the gradient. In the update rule, the fraction of the estimated mean and variance of the gradient is used instead of the raw gradient, whereby a parameter ϵ is introduced to avoid division by zero. Adam is robust and often computationally more efficient than other optimizers such as AdaGrad and stochastic gradient descent [23].

4.3. Backpropagation

Modern ANNs often consist of up to millions of trainable parameters, making the numerical differentiation of the cost function, which is required for e.g. gradient descent, very slow and inefficient. To tackle this problem, one first considers that

$C(x_{in}, \theta) = C(x_{in}, F_\theta(x_{in}))$ and makes use of the chain rule:

$$\frac{\partial C(x_{in}, \theta)}{\partial \theta_s} = \frac{\partial C(x_{in}, F_\theta(x_{in}))}{\partial F_\theta(x_{in})} \frac{\partial F_\theta(x_{in})}{\partial \theta_s} \tag{4.13}$$

On the level of neurons, this equation is given as

$$\frac{\partial C(x_{in}, \theta)}{\partial \theta_s} = \sum_g \frac{\partial [C(x_{in}, F_\theta(x_{in}))]_g}{\partial [F_\theta(x_{in})]_g} \frac{\partial [F_\theta(x_{in})]_g}{\partial \theta_s}, \tag{4.14}$$

where $[F_\theta(x_{in})]_g = [y_{out}]_g = a_{n_{out}}(z_g^{(n_{out})})$ denotes the value of neuron g in the output layer n_{out}. If θ_s is not a parameter of the output layer, the chain rule must be applied to $\partial [F_\theta(x_{in})]_g / \partial \theta_s$ and its derivatives until one arrives at the layer that contains parameter θ_s. For efficiency it is useful to order the parameters θ_s according to their respective layer, from the last to the first layer, because to calculate the derivatives of the cost function with respect to the parameters in layer $n-1$ one can reuse the derivatives with respect to the parameters of layer n due to the chain rule. Going through the layers from output to input and storing the derivatives per layer makes it possible to efficiently calculate all parameter updates $\delta\theta_s$ in one go. In other words, one propagates the deviation from output to target backwards (against the forward direction of the flow of information) and therefore this process is called *backpropagation*.

4.4. Convolutional layers

So far, only fully-connected layers have been introduced. However, constructing a *deep* ANN (ANN with many hidden layers) using only fully-connected layers leads to extremely large numbers of trainable parameters as the number of weights between two layers of N and M neurons is given by $N \cdot M$, casting fully-connected deep neural networks extremely slow. Alternatively, so-called *convolutional layers* can be used instead of fully-connected ones. The differences between the two are first that for convolutional layers, a neuron of layer n is only connected to a small set of neighboring neurons of layer $n-1$ and second that all neurons in a convolutional layer share the same weights. This is illustrated for the one-dimensional case in Figure 4.2. One exemplary neuron and the neurons it is connected to are marked in green. The three weights are called the *kernel* and are the same for all neurons in this layer. The number of weights in the kernel along one dimension is called kernel size and is a tuple for higher dimensions. Conventionally, a single integer kernel size k denotes a kernel, where the kernel size along all dimensions is k. A 2D convolution operation for 4×4 input and 2×2 output neurons and a kernel size of 3 is sketched in Figure 4.3a. As the name

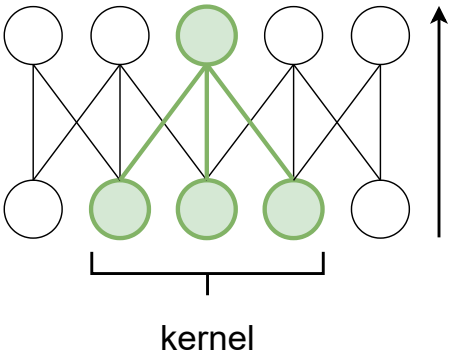
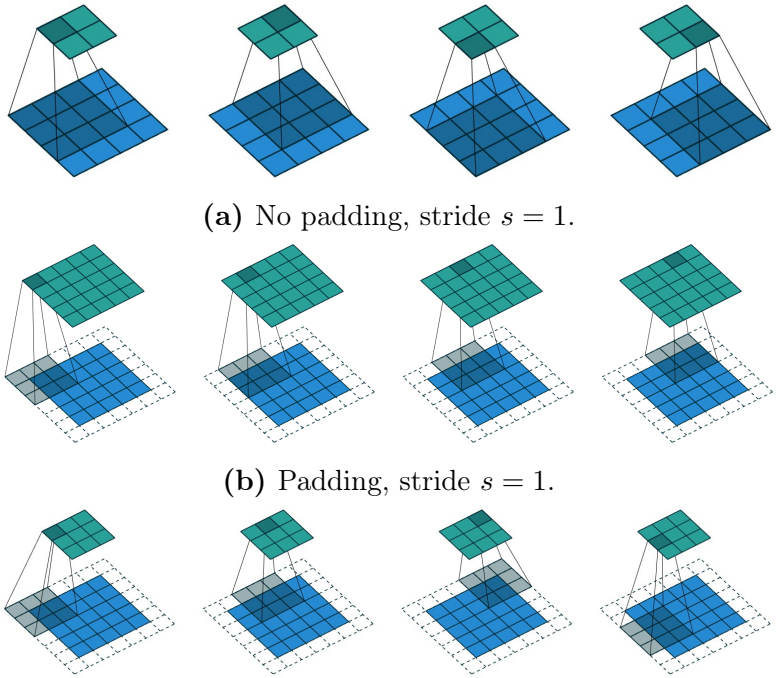


Figure 4.2.: Sketch of a 1D convolutional layer with a kernel size of 3. All neurons share the same kernel. The forward direction is here from bottom to top.



(a) No padding, stride $s = 1$.

(b) Padding, stride $s = 1$.

(c) Padding, stride $s = 2$

Figure 4.3.: Convolution operation of an a) (4×4) and b), c) (5×5) input with a (3×3) kernel with and without padding and different strides. The forward direction is here from bottom to top. Taken from [24].

suggests, convolutional layers perform a (discrete) convolution of the layer neurons with the kernel, where the boundaries of the layers can be padded e.g. with zeros to conserve the layer size (see Figure 4.3b). Since the weights per neuron are the same across a layer, if an input is shifted by several neurons, the output is also shifted but otherwise the same, allowing convolutional layers to be translationally invariant. Often a convolutional layer has multiple *channels* (or *filters*) with their own kernel weights and bias. In two dimensions these channels can be understood as the red, green, and blue channels in an RGB color image such that the image is described by a $(n_x \times n_y \times 3)$ matrix. For convolutional layers, channels have their own trainable kernel and bias and allow the extraction/processing of multiple features at the same time, which is why channels are sometimes called *feature maps*. In contrast to the spatial dimensions of the layer (e.g. image height and width), the channels of two convolutional layers are fully-connected. Thus, the total number of trainable parameters in a convolutional layer with n_{ch} channels, $(k_1 \times k_2)$ kernel and $n_{ch,\,input}$ input channels is given by:

$$n_{parameters} = (n_{ch} \cdot k_1 \cdot k_2 + 1) \cdot n_{ch,\,input} \tag{4.15}$$

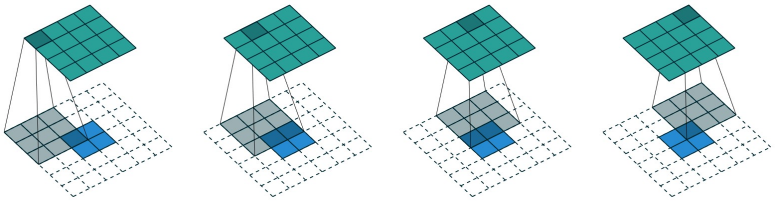
where the +1 comes from the bias that is trainable for each channel.

To decrease spacial dimensions using convolutional layers, one can use *strided convolutions*. The stride s denotes the distance between the neurons the kernel is applied to. For a stride of $s = 1$, the kernel is applied consecutively to all neurons, preserving spacial dimensions in combination with padding. In contrast, for a stride of $s = 2$, the kernel is applied to every second neuron, cutting even spacial dimensions in half. An example of a 2D convolution operation with padding and stride $s = 2$ is depicted in Figure 4.3c.

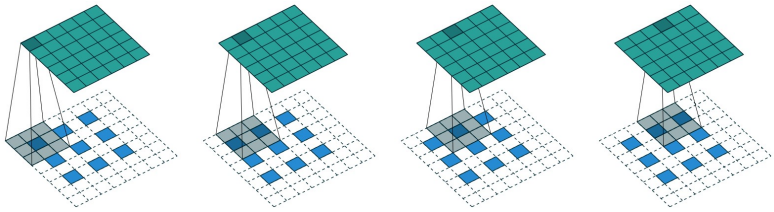
A method to increase spacial dimensions is *transposed convolutions*. The simplest way to increase dimensions is to use padding as shown in Figure 4.4a, however padding alone is not effective for larger layers. Another method sketched in Figure 4.4b is to choose a stride s and stretch the input into an intermediate grid with optional padding, such that the distance between neurons in the intermediate grid is s. Then, the convolution operation is applied to the intermediate grid with a stride $s_{grid} = 1$.

4.5. Training

As described above, the learnable parameters of an ANN are adjusted using an optimizer via backpropagation to minimize a loss function. This process is called



(a) Padding, stride $s = 1$.



(b) Padding, stride $s = 2$

Figure 4.4.: Transposed convolution operation of an a) (2×2) and b) (3×3) input with a (3×3) kernel with padding and different strides. The forward direction is here from bottom to top. Taken from [24]

training and ANNs are in this context often referred to as *models*. The set of training samples is divided into batches, where parameter updates are done per batch. A so-called *epoch* is completed after all samples were used for training once since the end of the last epoch or, in the case of the first epoch, the start of training.

Since a neural network consists of many adjustable parameters, *overfitting* is a common problem. Overfitting occurs if the model can produce accurate and meaningful predictions for the data it was trained on but fails to do so for new, unseen data. This indicates that the ANN is too biased towards the training data and has not learned to adapt to new data. To assess, whether overfitting occurs during training, one splits the available data into a *training set* and a *validation set*, where the size of both sets depends on the individual problem and the number of available data. The ANN is then exclusively trained on the training set, but at the end of each epoch, the loss of the validation set (validation loss) is evaluated. The behavior of both training and validation loss can then provide information about possible overfitting. Ideally, both validation and training loss should be similar in value, decreasing with an increasing number of epochs and eventually reach convergence if one trains long enough.

In reality, the model may develop some small bias towards the training set, resulting in a lower training loss. However, this is not a big problem as long as both training and validation loss are decreasing. In case of heavy overfitting, the validation loss might increase after some time while the training loss further decreases, since the model strengthens its bias towards the training data. This is sketched

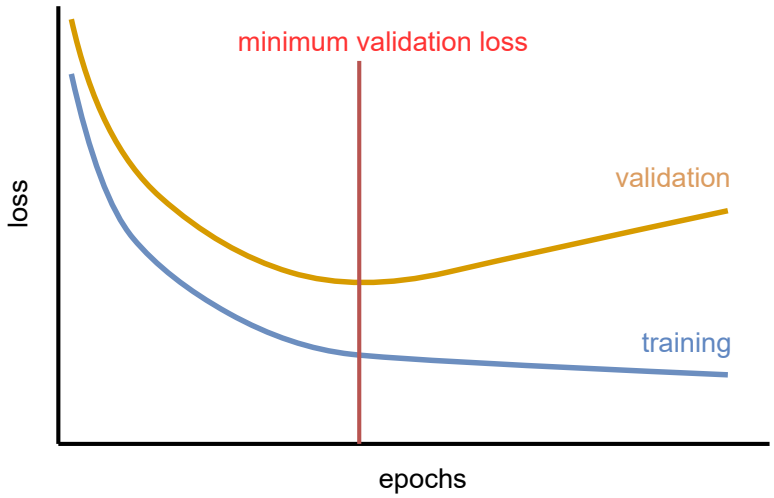


Figure 4.5.: Exemplary behavior of training and validation loss with overfitting, where the validation loss increases while the training loss decreases. The minimum validation loss is marked in red.

in Figure 4.5. Marked in red is the minimum of the validation loss. Since then the ANN experienced overfitting and the final model has significantly higher validation loss. To obtain the "best" model i.e. the model with the lowest validation loss, one can save checkpoints of the model (*model checkpointing*) after each epoch and then choose the one where the validation loss was lowest. Furthermore, to decrease unnecessary training time, one can end training if the validation loss has not decreased for a certain number of epochs. This is called *early stopping*. If the model with the lowest validation loss is selected, there may be a small bias towards the validation data. Hence, a third data set, the *test set*, is required if one wants to evaluate the model on completely unseen data.

In this thesis, the python frameworks `tensorflow`[25] and `keras`[26] (now part of `tensorflow`) are used to build and train ANNs. `keras` provides a variety of prebuilt layers, loss functions, and optimizers, among others the Adam optimizer that is used in this thesis. Furthermore, so-called *callbacks* that incorporate methods such as model checkpointing and early stopping are implemented.

4.6. Further aspects of machine learning

In this section, additional aspects of machine learning, such as regularization, hyperparameters and data preparation are described.

4.6.1. Regularization layers

To fight overfitting, one can make use of *regularization layers* such as *dropout* layers and *Gaussian noise* layers. The dropout layer randomly sets the value of a fraction of neurons to zero, whereby the fraction is called the dropout rate and can be chosen. This ensures that the model does not rely too heavily on certain neurons and learns to produce reasonable output using only a fraction of neurons. Gaussian noise layers add normal distributed noise to each neuron, where the variance can be chosen. This way, the model has to deal with small variations in the data and may learn to adapt to unseen data, reducing overfitting. It is important to note that the dropout and Gaussian noise layers do not change the shape of their input. For the `keras` implementation of these two layers, there are two modes, *training mode* and *inference mode*. During the training of the ANN, the training mode is active and the regularization layers operate as described above. The inference mode is active during the evaluation of the ANN after training, where the regularization layers are no longer needed. Therefore, a regularization layer in inference mode does absolutely nothing, i.e. its output is identical to its input.

4.6.2. Hyperparameters

Apart from the trainable parameters i.e. the weights and biases of each layer, there are other parameters such as layer size, number of channels of a convolutional layer, dropout rate, batch size, number of epochs, and learning rate. These so-called *hyperparameters* are not learned during training and have to be chosen before training. Still, they do have a significant impact on the training and specific combinations of hyperparameters can drastically improve training time, convergence, and model performance and may reduce overfitting. Therefore, tuning of these hyperparameters is often recommended. Since the search space can become very large, finding the optimal set of hyperparameters is difficult or almost impossible. However, automated tuning algorithms such as KerasTuner [27] can cover a considerable fraction of the search space and return the parameter set that came closest to the chosen objective. This, however, requires a well-defined objective, which is not always the case.

4.6.3. Data preparation and augmentation

Before training a neural network, it is important to prepare the training data, depending on the task and network structure. This includes preparing the data into a format so it can be used as input, normalizing and shifting data to lie in a desired interval, and reducing resolution to reduce network size and training time. Apart from that, operations such as rotation, magnification, and translation may be applied to copies of samples to increase the number of samples and introduce more

variation in the data. This is called *data augmentation*. However, augmentation is only recommended if it does not change the nature or the physics of the sample.

4.6.4. Supervised and unsupervised learning

So far, it has been assumed that the target output is always known and accessible. This scenario is called *supervised learning* since the ANN requires supervision (target output) to be trained. Depending on the specific task, the input is e.g. a set of images depicting various animals and the target output is a set of labels denoting the respective species. However, there are situations, where a target output is not known, accessible, or existent. Thus, one cannot train the model directly as described before, since a target output is required. This problem can be solved by *unsupervised learning*: One adapts the architecture of the ANN in such a way that it can do successful training using just the (unlabeled) training samples without any supervision (target outputs). An example of unsupervised learning is the *autoencoder* (see section 4.9).

4.6.5. Regression, classification and clustering tasks

Three of the major types of machine learning tasks are regression, classification, and clustering. Clustering tasks are those, where the model tries to find patterns or similarities in the data to group similar samples together. Depending on the algorithm, this is done with or without known labels. For classification tasks, the model is trained to predict a label (usually integers) for a given input and compares them to target labels. Lastly, regression tasks usually predict a real-valued continuous output that may be constrained to some interval, depending on the individual goal. The predicted output of the regression task can then be compared to a target, if available.

4.7. Hierarchical density-based spatial clustering of applications with noise

In this thesis, the clustering algorithm HDBSCAN (Hierarchical density-based spatial clustering of applications with noise)[28] is used to identify event clusters. This section follows the description of HDBSCAN in [29].

Compared to many other clustering algorithms, HDBSCAN does not require the number of clusters as an input parameter. Furthermore, HDBSCAN allows noisy data points i.e. data points not assigned to any cluster.

The HDBSCAN algorithm clusters points according to their density, which can be approximated using the so-called *core distance* d_{core}. For a *core point* x, $d_{core}(x)$ is the distance to its m_s nearest neighbors, where m_s denotes the minimal number of samples for x to be considered a core point. Using the core distance, the *mutual reachability distance* $d_{mreach}(a, b)$ of two points a and b is defined as the maximum over the core distances of the individual points and the metric distance between a and b. This d_{mreach} metric then spreads points in low-density regions further apart, while points in high-density regions are not affected. After that, the data is treated as a graph with d_{mreach} as weights on the edges between two points. From this graph, the *minimum spanning tree* is calculated, that is a graph consisting of only a subset of edges without any loops and minimal sum over all used edge weights. The tree edges are then sorted in increasing order with respect to their weight, creating a hierarchy. Removing all edges above a certain chosen threshold on the edge weight would split the minimum spanning tree into smaller disconnected sub-graphs and sometimes individual points.

This alone could already provide a clustering, however, the threshold is not easy to choose intuitively. HDBSCAN solves this by introducing a minimum cluster size m_{cl}. While steadily lowering the threshold on the edge weights, a parent cluster is at some point split into two child clusters. If one of the child clusters contains fewer points than m_{cl}, the split is not considered a *proper* split and the smaller child cluster is not considered anymore for lower thresholds. On the other hand, if both child clusters contain at least m_{cl} points, the split is a proper split and the child clusters are considered for lower thresholds until the next split appears. HDBSCAN then introduces a *cluster stability*, which measures how persistent a cluster is with respect to its size and how long it lives with respect to the threshold on the edge weights. Then, going from low to high thresholds, if the sum of the stabilities of child clusters at a proper split is higher than the stability of the parent cluster, the stability of the parent cluster is set to the sum of the stabilities of its child clusters and one proceeds to the next proper split. On the other hand, if the sum of the stabilities of the child clusters is lower than the stability of the parent cluster, the parent cluster is considered a proper cluster. The procedure is finished until one arrives at the first proper split at the highest threshold. The final clustering consists then of all proper clusters and all remaining points not assigned to any proper cluster are called *unclustered*.

In this thesis, the HDBSCAN implementation `hdbscan` [30] is used. Here, m_s is controlled by `min_samples` and m_{cl} is adjusted by `min_cluster_size`.

4.8. Uniform Manifold Approximation and Projection

To represent high-dimensional data points in a low-dimensional space, one can use dimension reduction algorithms. In this thesis, the algorithm UMAP (Uniform Manifold Approximation and Projection) [31] is used to visualize high-dimensional data in a two-dimensional space. One advantage of UMAP over some other algorithms is that it tries to preserve global structure [31].

Given a set of high-dimensional data points, UMAP first constructs a graph in this high-dimensional space, where per point only its $n_{neighbors}$ nearest neighbors are considered. Then the distances between connected points are calculated according to some metric. These distances are then used to compute a similarity score of one point to another point. These scores are then used as weights for the graph edges. After that, a low-dimensional embedding is initialized. The goal is now to optimize this embedding with respect to an objective function so that the relations between points in the high-dimensional graph resemble the low-dimensional embedding. This can be done using stochastic gradient descent with a certain number of optimization steps.

In this thesis, the UMAP implementation `umap` [32] is used, where the parameter $n_{neighbors}$ is called `n_neighbors` and the number of dimensions of the low-dimensional embedding is controlled by `n_components`.

4.9. Autoencoders

In this part, an unsupervised artificial neural network architecture, the *autoencoder* (AE), and its applications are introduced.

4.9.1. Autoencoder structure and feature extraction

Consider having access to a set of unlabeled training samples e.g. grayscale images each containing a square in white on a black background. The image is then fully described by four parameters: position (x and y), side length, and rotation. A human is most likely able to determine these parameters. But it is also possible to extract the parameters using an ANN. A suitable architecture is the autoencoder. Its task is to produce an output as close as possible to the original input. The trivial way to solve this problem is to simply copy the input. However, one does not gain any useful information about the properties of the data. The idea is to force the autoencoder to extract the most important features (here ideally side length, rotation, and position of the square) of the data by introducing a *bottleneck*

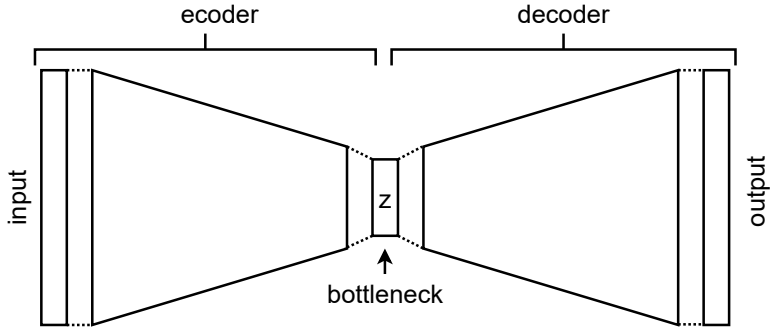


Figure 4.6.: Structure of an autoencoder. The input is compressed by the encoder into a lower-dimensional representation, the bottleneck z, which is then passed to the decoder that produces an output.

layer between the input and output layer. Figure 4.6 shows the structure of such an autoencoder, which consists of two ANNs, *encoder* $E(x)$ and *decoder* $D(x)$. The bottleneck layer z is the output layer of the encoder and the input layer of the decoder. The input of the encoder and output of the decoder have the same shape and format (equal to that of the images). The task of the autoencoder $F(x)$ is then given by

$$F(x_{in}) = D(z) = D(E(x_{in})) = y_{out} \overset{!}{\approx} x_{in}. \tag{4.16}$$

Here input and target are known and one can train the AE as described in section 4.2 and 4.3 by minimizing a loss function. In the context of autoencoders, the loss function, i.e. the error between input and output, is also called *reconstruction loss*. Since one wants the autoencoder to extract the most important features of the data, one chooses the number of neurons in the bottleneck layer to be (significantly) less than the number of neurons in the input layer (e.g. the number of pixels in the image). In this case, one speaks of an *undercomplete autoencoder*. Thus, the autoencoder is forced to compress the data into a lower-dimensional space in such a way that the decompression yields an output close to the input. As a result, autoencoders are also used for *dimensionality reduction* of high-dimensional data. In this context, the neurons in the bottleneck are often referred to as *latent variables*, where latent means that the variables are hidden and can not be inferred directly from the data. The low-dimensional space in which the data is represented is often called *latent space*. In the example from above one knows that the data can be fully described by four parameters (three, if one chooses the position to be the index of the flattened image). Therefore, following the principle of Occam's razor, one would choose the number of neurons in the bottleneck to be four, i.e. the minimum number of parameters to describe the data. For more complex prob-

lems, the ideal number of parameters is often not known and has to be estimated or guessed. If the bottleneck size is too small the autoencoder may not be able to accurately reproduce the input data. On the other hand, choosing the bottleneck size too large may tempt the autoencoder into simply copying parts of the input that contribute most to the value of the cost function and not learning the features of the data. For more complex and abstract data, assessing and understanding the learned features may become difficult.

4.9.2. Denoising autoencoders

Now, consider the case, where there is random Gaussian noise on the images from the example above. Since the noise is randomly distributed and the noise in one pixel is unrelated to the noise in neighboring pixels, it is not possible to compress the images without loss of information. However, if the noise is not too large compared to the noiseless images, the autoencoder may still be able to extract the features of the images. As the autoencoder cannot encode the noise, the output will likely be a cleaned version of the noisy input, potentially plus some offset to account for the mean of the noise. Thus, autoencoders can also be used to *denoise* noisy data.

4.9.3. Anomaly detection

In the sections above, three applications of autoencoders were introduced: feature extraction, dimensionality reduction, and denoising. Another application that will be discussed in the following section is *anomaly detection* [33].

Consider having access to a set of data from a detector containing samples from multiple event classes. However, it is not known whether the set contains all possible event classes. Furthermore, one is interested in selecting only one specific class from newly added unlabeled data. This problem can be solved using an autoencoder. The idea is to train an undercomplete autoencoder exclusively on data from the desired class. Ideally, if trained correctly, the autoencoder will then learn to compress and decompress the data of this class in such a way that the reconstruction loss is minimized, while the other classes remain unknown (anomalous) to the AE. That way, the AE has never learned to encode and decode data of other classes and hence it is not able to accurately reconstruct samples of other classes, resulting in a higher reconstruction loss for samples of unseen classes. By choosing a reconstruction loss threshold and discarding samples that exceed this threshold, it is possible to separate data of the desired class from the remaining classes. The choice of threshold usually depends on the chosen metrics and can be adjusted for individual analyses. Note, that the anomaly detection method only works if the

samples of the target class are sufficiently different from samples of other classes. Furthermore, a high noise level in the data of the target class may negatively impact the separation. If that is the case, preparing the data in such a way that the impact of noise on the reconstruction loss is minimized is recommended. For example, if the noiseless signal is stronger than noise, using the square of the noisy signal should suppress noise compared to the noiseless signal. The separation performance of this method also heavily depends on the purity of the training data and the size of the bottleneck. If the training set contains sufficiently many events from other classes and there are some free parameters in the bottleneck (if it is chosen too large), the autoencoder might learn to reconstruct those event classes as well. In the worst case, if the autoencoder has too many free parameters, the training data is very impure and the classes are sufficiently similar, no (or weak) separation may occur.

4.10. Variational autoencoders

This section follows the original article that first introduced *variational autoencoders* (VAEs) in 2013 [34]. First, a short motivation of VAEs is presented and after that, the mathematical concepts of VAEs are derived in more detail.

Standard autoencoders are deterministic neural networks, where the encoder maps an input sample to a single point in the latent space. This way, the latent space of an autoencoder is not necessarily organized because samples with completely different properties may be close to each other in the latent space. This becomes a problem, if one would like to use the latent space to either cluster similar samples into groups or generate new data by sampling from the latent space. Hence, an organized latent space is necessary for clustering and sampling tasks. A VAE tries to achieve an organized latent space by encoding samples as probability distributions rather than deterministic points. Since probability distributions have a variance, a sample is not always mapped to the same point in the latent space. This forces the VAE to organize the latent space in such a way that similar samples are closer in latent space and dissimilar samples are far apart.

With the goal of sampling new data from the latent space in mind, consider the likelihood $p_\theta(x)$ of some data x given (trainable) parameters θ. $p_\theta(x)$ is here assumed to be a Gaussian $\mathcal{N}(\mu, \sigma)$ that can be expressed by mean μ and standard deviation σ. The goal is to sample data x from some random variable z (the latent variable), hence one can write

$$p_\theta(x) = \int p_\theta(x, z)\,\mathrm{d}z = \int p_\theta(x|z)p_\theta(z)\,\mathrm{d}z. \tag{4.17}$$

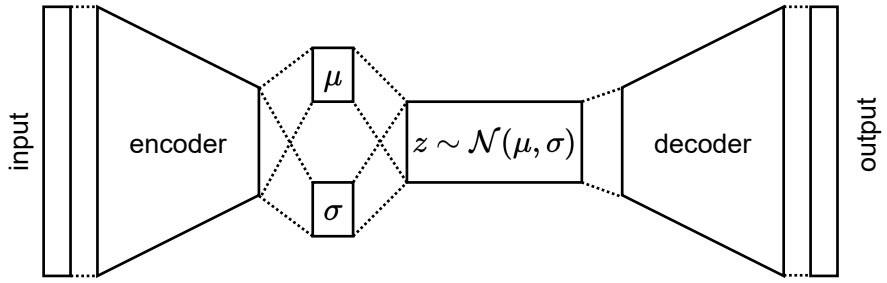


Figure 4.7.: Structure of a variational autoencoder. The encoder returns a mean μ and standard deviation σ that is then used to sample the latent variable z from a corresponding Gaussian distribution. z is then given as input to the decoder.

Here, $p_\theta(x, z)$ denotes the joint probability distribution of x and z, $p_\theta(x|z)$ is the conditional distribution of x given z and $p_\theta(z)$ denotes the probability distribution of z. Using Bayes' theorem, the *posterior distribution* is

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}, \tag{4.18}$$

that is the probability distribution of z given x. In this context, $p_\theta(z)$ is called the *prior distribution*. $p_\theta(z|x)$ is often not accessible and is approximated by the conditional distribution $q_\phi(z|x)$.

The original goal of variational autoencoders is that the decoding part learns to generate new data, similar to data x by sampling from a random variable z. Thus, the encoder has to learn how to represent the data x into the latent space z. Therefore, the encoder is expressed by the approximate posterior distribution $q_\phi(z|x)$ and the decoder is represented by the likelihood $p_\theta(x|z)$. The prior distribution $p_\theta(z)$ of the latent space is here chosen to be a multivariate Gaussian $\mathcal{N}(\mu = 0, \sigma = 1)$ and from now on denoted by $p(z)$, omitting the θ. The encoder, which is also assumed to be a multivariate Gaussian $\mathcal{N}(\mu, \sigma)$, is implemented such that it returns two layers, one for μ and one for σ. z is then sampled from $\mathcal{N}(\mu, \sigma)$ by a *sampling layer* and given as input to the decoder. The structure of such a variational autoencoder is depicted in Figure 4.7.

Variational autoencoders are trained by maximizing the *variational lower bound*, also called the *evidence lower bound* (ELBO), on the marginal log-likelihood of the data [34–36]:

$$\mathcal{L}(\theta, \phi; x) = -D_{KL}(q\phi(z|x)||p(z)) + \mathbb{E}_{q_\phi(z|x)}\left[\ln p_\theta(x|z)\right] \tag{4.19}$$

The second right-hand side term is the expectation value of $\ln p_\theta(x|z)$ with respect

to $q_\phi(z|x)$ which can be interpreted as a reconstruction likelihood and its negative is interpreted as a reconstruction error commonly implemented by a regression loss function, usually either the mean squared error or mean absolute error. The first right-hand side term is the *Kullback-Leibler divergence* (KL divergence) that measures the difference between the posterior distribution and the prior distribution [34]:

$$D_{KL}(q_\phi(z|x)||p(z)) = \int q_\phi(z|x) \ln\left(\frac{q_\phi(z|x)}{p(z)}\right) \mathrm{d}z \tag{4.20}$$

As noted above, both prior and posterior distributions are assumed to be multivariate Gaussians:

$$q_\phi(z|x) = \mathcal{N}(\mu, \sigma) \tag{4.21}$$
$$p(z) = \mathcal{N}(0, 1) \tag{4.22}$$

In this case, the KL divergence (often called KL loss) is given by [34, 36]:

$$D_{KL}(q_\phi(z|x)||p(z)) = -\frac{1}{2}\left(1 - \sigma^2 - \mu^2 + \ln\left(\sigma^2\right)\right) \tag{4.23}$$

The loss function of a standard variational autoencoder can then be written as

$$\text{total loss} = \text{reconstruction loss} + D_{KL}(q_\phi(z|x)||p(z)). \tag{4.24}$$

Note that the objective of the VAE is to minimize the loss function, hence the plus sign is required in front of the KL divergence in contrast to the minus sign in the ELBO in Equation 4.19 which should be maximized.

Usually, the magnitude of the loss differs for different reconstruction loss functions e.g. if the input and output are constrained to $[0, 1]$, the mean squared error is less or equal to the mean absolute error. Therefore, it is useful to introduce a weight β for the KL loss to be able to increase or decrease the influence of the KL divergence on the total loss:

$$\text{total loss} = \text{reconstruction loss} + \beta \cdot D_{KL}(q_\phi(z|x)||p(z)). \tag{4.25}$$

Such a variational autoencoder is also called β-VAE [37].

So far, the latent variable z is sampled from the posterior distribution. This, however, is a problem as there is the probabilistic layer z between the encoder and decoder, which does not allow to differentiate a loss function and use backpropagation for training. This problem is solved by the *reparameterization trick* [34]:

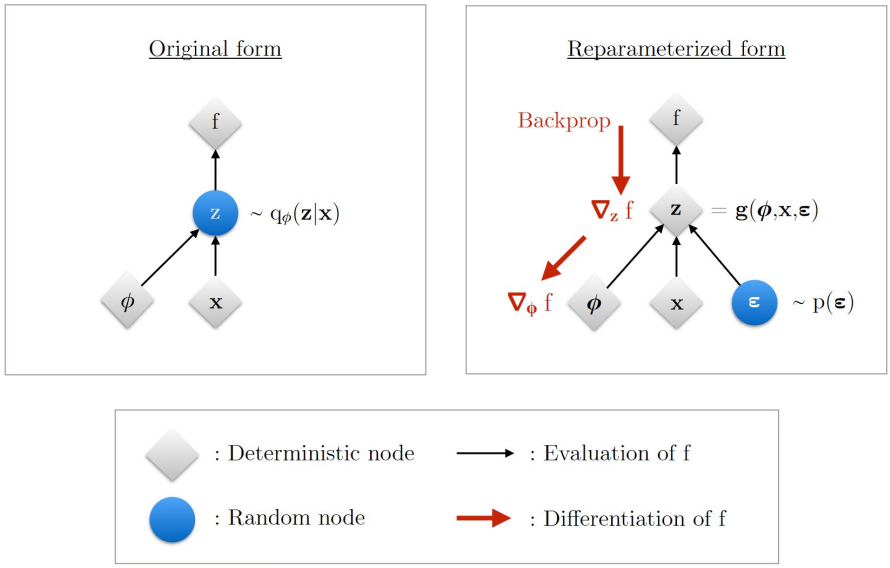


Figure 4.8.: Scheme of the reparameterization trick. In the original form (left), z is a random node and one cannot use backpropagation. By reparameterization, the random element is outsourced into a random number generator ϵ allowing backpropagation. Taken from [35].

Instead of sampling z from a distribution $q_\phi(z|x)$ (here Gaussian $\mathcal{N}(\mu, \sigma)$), one can use a random number generator to compute $\epsilon \sim \mathcal{N}(0, 1)$ and express z via

$$z = g(\phi, x, \epsilon) = \mu + \sigma \cdot \epsilon. \tag{4.26}$$

That way, as sketched in Figure 4.8, the loss becomes differentiable, since μ and σ, which depend on trainable parameters, are deterministic and backpropagation is possible again.

Instead of μ and σ or σ^2, one can choose the encoder to output μ and $\ln(\sigma^2)$. This ensures that the variance is positive and non-zero. The latent variable z is then given by

$$z = \mu + \sigma \cdot \epsilon = \mu + e^{\frac{1}{2}\ln(\sigma^2)} \cdot \epsilon. \tag{4.27}$$

5. Investigation of wind-correlated events

In this chapter, events recorded with RNO-G that appear during high wind speed periods are examined. First, shallow trigger rates of station 23 are investigated and compared to wind speed. Afterwards, a machine learning method is used to cluster wind events to analyze potentially wind-induced event classes and obtain a data set to train a discriminator to reject this background (see chapter 6).

5.1. Weather data

The weather data used for this thesis is provided by NOAA[38] and is measured at the main building at Summit Station ("Big House", SUM), up to several kilometers away from the RNO-G stations. Since the terrain at Summit is mostly flat, it is assumed that the weather is the same within a few kilometers around Summit Station and the RNO-G stations. The official NOAA files include minute averages for wind speed in m/s, temperature in °C at 2 m above ground, wind direction in degrees (zero is true north), barometric pressure in hPa, and relative humidity in %. The data is given for UTC time.

5.2. Investigation of station 23

To investigate wind events, station 23 is selected, since, compared to stations 11, 12, 21, and 22, it should not show strong battery charging (21, 22), LoRaWAN (21, 22), and wind turbine (11, 12) noise. Furthermore, out of the remaining stations (13, 23, 24), station 23 has the lowest trigger threshold, which increases trigger rates and allows to study lower amplitude wind events at the cost of having more thermal noise events that coincidentally met the trigger condition. Data-taking for station 23 began in the last half of 2022 and continued in the spring of 2023. However, due to some unknown reason, shallow trigger rates of 2023 show a significant and constant increase compared to 2022, although no changes to the trigger settings were made. As a result, this investigation is restricted to 2022 data. Due to satellite transfer rate limitations, only the sub-sampled data is used for this

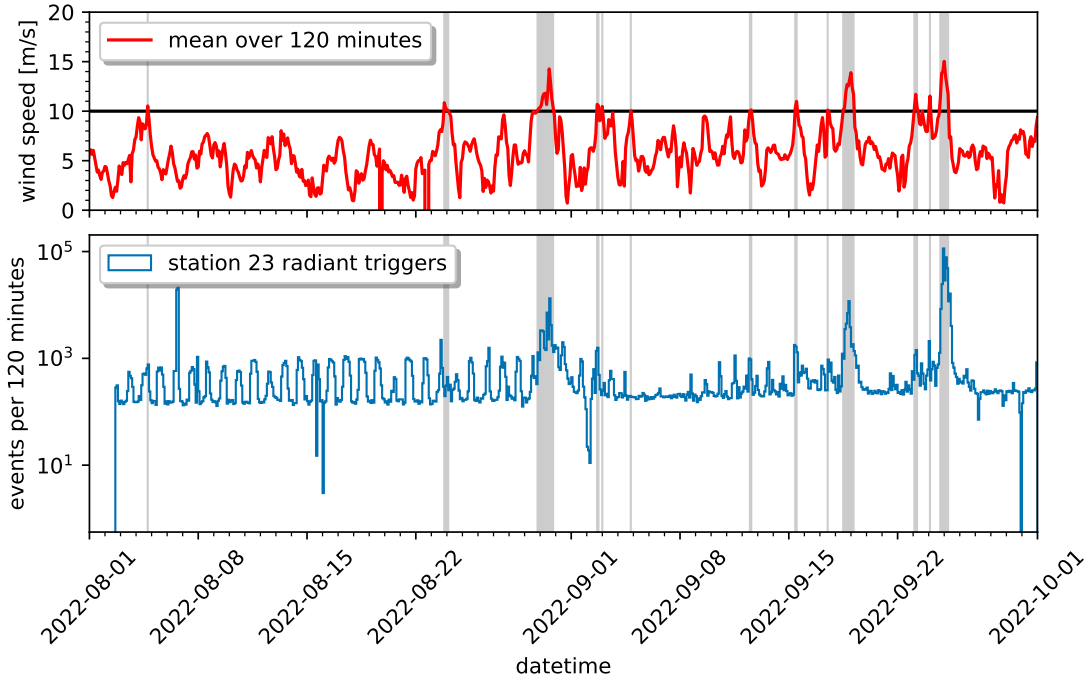


Figure 5.1.: Station 23: shallow (radiant) trigger rates (in counts per 120 minutes, logarithmic scale) as well as the mean wind speed over 120 minutes. Time intervals where the mean wind speed is greater or equal to 10 m/s (marked by a horizontal line) are shaded in gray. Time is given in UTC.

analysis. An exception are trigger rates, which are available for the full data set.

In 2022 station 23 took data from the end of June to the beginning of October. However, during the first month, shallow trigger rates are very unstable (see Figure A.1 in appendix), since suitable trigger settings had to be found first. Thus, only the time period from August to October 2022, where the trigger rates are stable, is selected. Furthermore, only those runs that behaved as usual, i.e. runs that ran for the full two hours, are considered and shorter runs, including calibration runs, are excluded.

Figure 5.1 shows the radiant (shallow) trigger rate (full data from the header files) of station 23 between 2022-08-01 and 2022-10-01 as well as the mean wind speed $\langle v_{wind} \rangle$. In [7], a turn-on wind speed of roughly 10 m/s was observed for RNO-G, hence the time intervals where the average wind speed was greater or equal 10 m/s is shaded in gray for better visualization. In most of the cases, a strong increase in the event rate coincides with $\langle v_{wind} \rangle \geq 10$ m/s. However, trigger rates do not always increase if the wind speed crosses the threshold (see 2022-

08-05). Another interesting aspect is that on 2022-09-25 the trigger rate is one order of magnitude higher compared to 2022-08-30 and 2022-09-19, although the average wind speeds only differ by $\sim 1\,\mathrm{m/s}$. Furthermore, an afterglow-like effect can be observed where for very-high-wind days, event rates, although decreasing, are still above baseline even a few hours after $\langle v_{wind} \rangle$ has fallen below the $10\,\mathrm{m/s}$ threshold. Noticeable is also a periodic behavior of the trigger rate during August, where the trigger rates exhibit a roughly constant increase exclusively during the day. This behavior may be connected to the charging of solar panels, which happens exclusively during daytime and becomes weaker towards the end of the year when the sun is lower. Apart from the solar panels, human activity could also be responsible for this behavior. Although the source of this behavior is not known, it shows that there are still other event classes apart from wind and CW present in the data. Therefore, to obtain a high-purity data set of wind events, wind events must be separated from other backgrounds and one can not rely solely on the wind speed at the time of the event.

5.3. Event clustering using a variational autoencoder

As described above, other backgrounds are present in station 23 data and thus events that appear during high-wind periods are not necessarily wind-induced. Furthermore, event rates stay high after the wind speed has dropped, indicating that wind-correlated signals can under some circumstances appear at lower wind speeds. Therefore, a method to group events of the same class together is required. For this thesis, a machine learning clustering method inspired by [39] was chosen and is presented in the following.

5.3.1. Strategy

The goal is to find similarities and differences between events to group similar events into clusters. The trivial solution that results in the highest purity is to assign each event its own cluster, but one does not gain anything by doing that. Therefore, the number of clusters should be much smaller than the number of events. There are already numerous clustering algorithms available but these methods usually require a vector of features as input. It is possible to use the raw waveforms as input, but the individual samples do not resemble the features of the waveform, especially since the signal can be shifted in time. Thus, one first has to compress event data into a set of features to effectively use a clustering algorithm. As described in section 4.9, autoencoders and especially variational autoencoders can be used to compress high-dimensional data into a vector of features i.e. the

bottleneck or latent space. Furthermore, the clustering algorithm HDBSCAN is chosen as it is fast, does not require a fixed number of clusters as input, and allows outliers i.e. events not assigned to any cluster. Variational autoencoders are expected to work well, especially in combination with density-based clustering algorithms such as HDBSCAN, since the latent space is modeled according to a chosen probability density function. For an RNO-G station S, the strategy is then to

1. train a variational autoencoder on station S data (optimize hyperparameters)
2. for all station S events: compute the mean latent space vector (μ in Equation 4.27) instead of the sampled latent variable z to obtain a deterministic clustering
3. obtain clusters using HDBSCAN with the mean latent space vectors (μ in Figure 4.7) as input (target high clustering efficiency) and visualize results using UMAP
4. assess clusters by inspecting individual events and investigating wind speed distribution per cluster
5. optimize HDBSCAN parameters or adjust the size of the bottleneck if clusters are not meaningful

5.3.2. Data preparation

As input, the square of the short-time Fourier transform (STFT) magnitude (also known as spectrogram) of the waveform is chosen. To obtain the STFT, the waveform is divided into time intervals using a window function, where the intervals can overlap. Then, the frequency spectrum per interval is computed using the Fast Fourier Transform (FFT). In the end, one obtains a frequency spectrum as a function of time, which can be represented by a 2D matrix with frequency on one axis and time on the other axis. One advantage of the STFT is that the STFT makes it easy to determine at which time and frequency the signal has considerable power. The 2D matrix format makes it very suitable for machine learning, as both time and frequency information can be processed in the same layer. Furthermore, thermal noise is suppressed compared to the actual signal waveform, since thermal noise has a flat frequency spectrum at all times, leading to a baseline offset with random fluctuation in the STFT. The element-wise square of the STFT magnitude is used in this analysis to further suppress the influence of thermal noise. Due to the trigger, the signal appears always in the second half of the waveform, hence the first part is not used. The STFT magnitude is calculated using the `scipy` [40] function `scipy.signal.stft`. The segment size can be controlled by `nperseg`

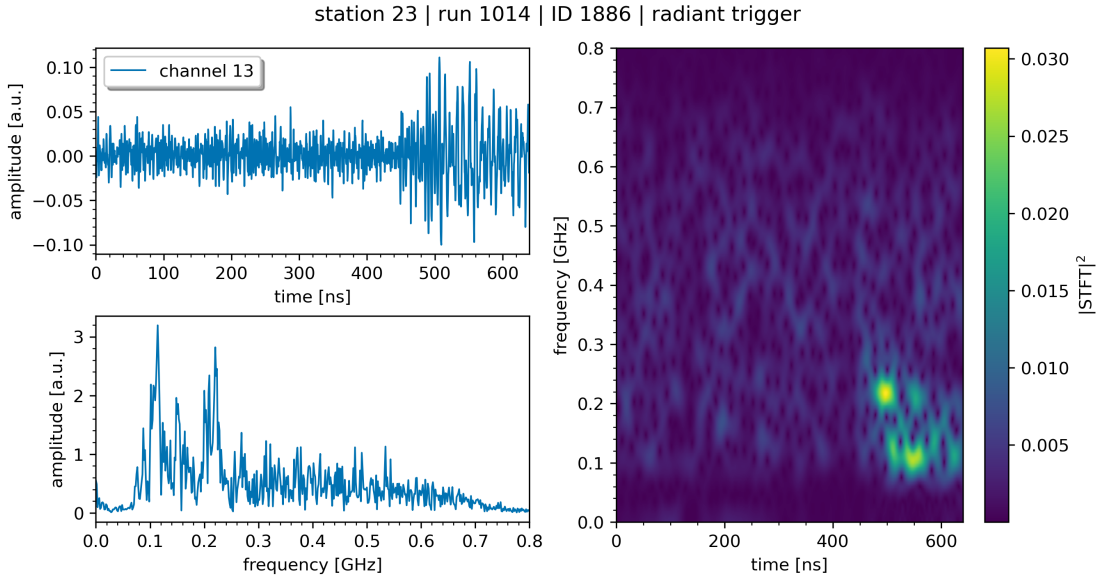


Figure 5.2.: Waveform, frequency spectrum and spectrogram ($|\mathrm{STFT}|^2$) of an example event recorded with channel 13 of station 23.

and the overlap between segments by `noverlap`. Small segment sizes give low resolution in the frequency domain and high resolution in the time domain and vice versa for large segment sizes, however, it is possible to artificially increase the number of frequency bins using the `nfft` parameter. Increasing `nfft` leads to more smoothness in the frequency domain but does not increase resolution. In this thesis the following values are chosen: `nperseg` = 128, `noverlap` = `nperseg` − 1 and `nfft` = 2048. An example of the spectrogram for a waveform and corresponding frequency spectrum is shown in Figure 5.2.

Furthermore, only the first half of the STFT in the frequency domain is used, as this represents the frequencies from 0 Hz to 800 MHz where the signal is amplified. The resulting STFT is then downsampled to 128×128 pixels. Per event, all three upward-facing LPDAs are used and their STFT magnitudes are stacked into a $128 \times 128 \times 3$ matrix. This matrix is then squared element-wise, normalized to $[0, 1]$, and converted into 32 bit floats to be used as a `tensorflow` input. The normalization is an element-wise division by the maximum of all elements of the matrix.

Usually in machine learning, one can use data augmentation to generate new data by modifying existing data, e.g. by rotating, magnifying, or shifting the image. However, in this case, rotation and magnification are not meaningful as this would change the physical properties of the data. Although translation is a physically valid and meaningful operation, translation is disfavoured, since the waveforms are often clipped at the end. To still have some variation in the data, a Gaussian

noise layer is included directly after the input of the VAE as well as several dropout layers for regularization.

For the training and validation data, only shallow-triggered events from August to October of 2022 were chosen, since wind events are assumed to be produced near the surface. The force triggers are excluded from training since they include mostly thermal noise, which is random and cannot be compressed into a few parameters in a meaningful way. Depending on the trigger threshold, thermal noise can occasionally trigger an event, hence events with a weak signal are not desired for training. Therefore, the *signal-to-noise ratio* (SNR) is introduced, where the SNR for a noisy waveform x and noise r is defined as

$$\mathrm{SNR}(x) = \frac{\max(|x|)}{\mathrm{RMS}(r)}, \quad \mathrm{RMS}(r) = \sqrt{\langle r^2 \rangle} \tag{5.1}$$

RMS denotes here the *root mean square*. The noise RMS can be obtained by taking the median over many force trigger events, which are mostly pure thermal noise. The median is here chosen over the mean to reduce the impact of rare outlier events, such as CW signals that may appear coincidentally in force trigger events. For station 23, a median SNR of 3.8 is obtained for force-triggered events. To filter out weak signals and thermal noise from the shallow triggered events, an SNR cut is applied, where events must satisfy SNR$\geq$ 5 in two out of three upward-facing channels.

5.3.3. Model architecture

The variational autoencoder is implemented using `tensorflow` and `keras`. The basic framework of the model is adapted from [41] and the structure of the encoder (decoder) is shown in Table A.1 (Table A.2). The input layer just handles the input data and sends it to the next layer, a Gaussian noise layer that adds some variation to the data. After that, there are a series of convolutional layer blocks consisting of two 2D convolutional layers (Conv2D in `keras`), activation functions (here LeakyReLU with $\alpha = 0.3$), and a dropout layer (except for the last block). In general, the strategy is to decrease layer size along spatial dimensions (n_x, n_y) and increase the number of channels (filters) n_{ch} to extract features, where the output shape of such a 2D convolutional layer is (n_x, n_y, n_{ch}). The first convolutional layer in each block increases the number of channels (filters) by a factor of 2 starting from 16 and uses a kernel size of 1 while the second convolutional layer reduces spatial dimension by a factor of 2 using stride $s = 2$ and kernel size 3, except for the first block where the kernel size is 5. By separating the increase in channels and decrease in spatial dimensions over two layers, the network becomes deeper and an additional non-linearity (activation) is introduced between these

operations. The dropout rate is here 0.3 for all blocks. The output of the last convolutional layer is flattened into a vector and then passed to an intermediate fully-connected layer (Dense in `keras`) that is connected to the mean and log-variance latent space layers of length 9, both fully-connected. The latent variable z is then computed in the sampling layer according to Equation 4.27. The decoder is mostly a mirrored version of the encoder without sampling, except that 2D convolutions are replaced by transposed 2D convolutions (Conv2DTranspose) and the intermediate fully-connected layer is missing to reduce the number of trainable parameters in the decoder. The output of the decoder is a 2D convolutional layer with three channels and a ReLU activation that is limited to $[0, 1]$ to match the input data.

5.3.4. Results for station 23

Now, the training of the VAE and clustering results for station 23 are presented.

5.3.4.1. Training

The data is split into 70 % training and 30 % validation data without test data as the loss is not evaluated after training since only the encoder is of interest. For the reconstruction loss function, the mean absolute error (mean over batch) is chosen with a weight $\beta = 0.99$ on the KL divergence. The Adam optimizer is used with a learning rate of $\eta = 2 \times 10^{-4}$ and otherwise default parameters. The model is then trained for at most 200 epochs with a batch size of 16, model checkpointing, and early stopping. During training the learning rate is reduced according to $\eta \to 0.3\eta$ if the total validation loss has not decreased for 10 epochs. In the end, the model with the minimal total validation loss is saved. Training and validation loss for the VAE trained on station 23 data are shown in Figure 5.3. Both validation and training loss show very similar decreasing behavior with the training loss being slightly lower than the validation loss as one would expect. The sudden drop in the loss after epoch 125 is presumably due to a change in learning rate. The minimum validation loss is marked by a red vertical line. After that, the validation loss does not decrease but it does not increase noticeably either.

From first clustering attempts it was observed that for station 23 there is one cluster that contains substantially more events than other clusters and appears only during one specific high-wind day. To reduce potential bias and focus of the variational autoencoder on this dominant cluster, three data runs (1015, 1016, 1017) are excluded from the final training of the VAE but are still used for the analysis.

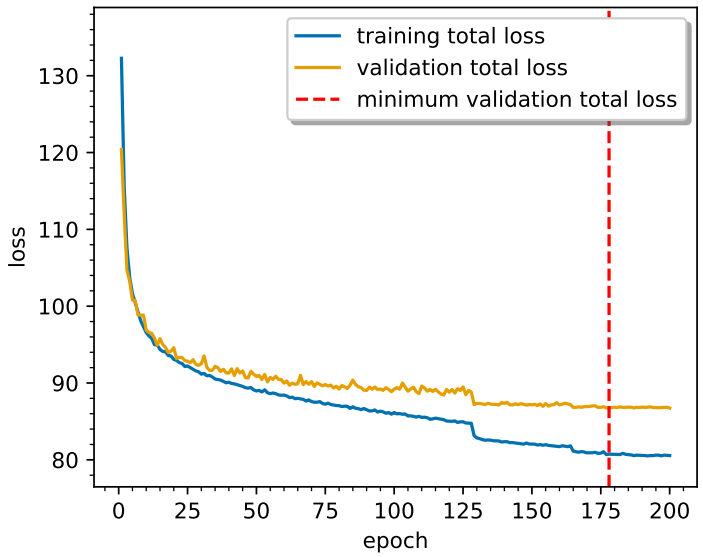


Figure 5.3.: Total loss of the variational autoencoder trained on station 23 data for event clustering. The reconstruction loss is the mean absolute error.

5.3.4.2. Clustering results

After training of the VAE, the mean latent vector (directly from the `z_mean` layer) is computed for all shallow- and force-triggered events in the investigated time period. The vectors are then given as input to the HDBSCAN clustering algorithm. After that, the minimal cluster size (`min_cluster_size`) as well as the parameter `cluster_selection_epsilon` is tweaked while the resulting clusters are manually investigated to get an accurate and efficient clustering. `cluster_selection_epsilon` controls here how close clusters must be to combine them into one cluster. The parameter `min_samples` is set low so that more data points are core points and fewer data points are classified as noise in the end. Furthermore, the parameter `cluster_selection_method` is set to "leaf", which returns more fine-grained clusters [30].

To qualitatively assess clustering accuracy and efficiency, the bottleneck vectors are represented in a two-dimensional space using the dimensionality reduction algorithm UMAP that allows to visualize the data in a scatter plot and mark the clusters obtained by HDBSCAN. It is then possible to select an individual point in that two-dimensional space and investigate the corresponding event including its waveform, spectrum, and STFT per channel. Since UMAP tries to conserve both global and local structure, events of the same class should lie close together. To assess the clusters, it is first investigated whether clusters found by HDBSCAN

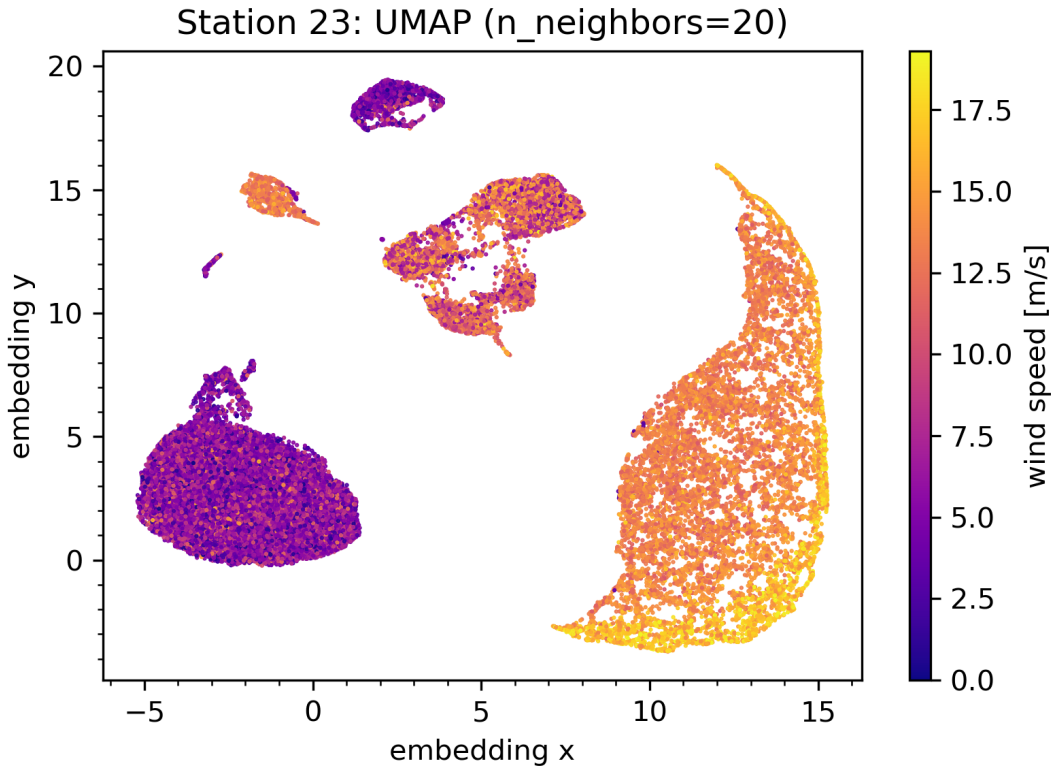


Figure 5.4.: UMAP representation of the mean latent vectors of station 23 shallow- and force-triggered events with the corresponding wind speed encoded by color.

do cluster in the UMAP embedding as well. After that, individual events of each cluster are inspected in terms of waveform, spectrum, and STFT. Then the borders of each cluster are examined and events that are near a cluster but not assigned to any cluster are investigated to judge clustering efficiency. If some clusters are split in the UMAP and contain more than one visually different event class, the HDBSCAN parameters are tweaked again. The `min_cluster_size` is kept low to allow small clusters and the `cluster_selection_epsilon` is slowly altered to combine close clusters if their events are visually very similar to avoid having hundreds of small clusters.

For station 23 the UMAP representation with wind speed encoded by color is shown in Figure 5.4 and with the final clustering in Figure 5.5. In the appendix, the event density in the UMAP representation is depicted in Figure A.4 and the final non-default HDBSCAN parameters are found in Table A.3.

At first, the UMAP representation without the clusters found by HDBSCAN is

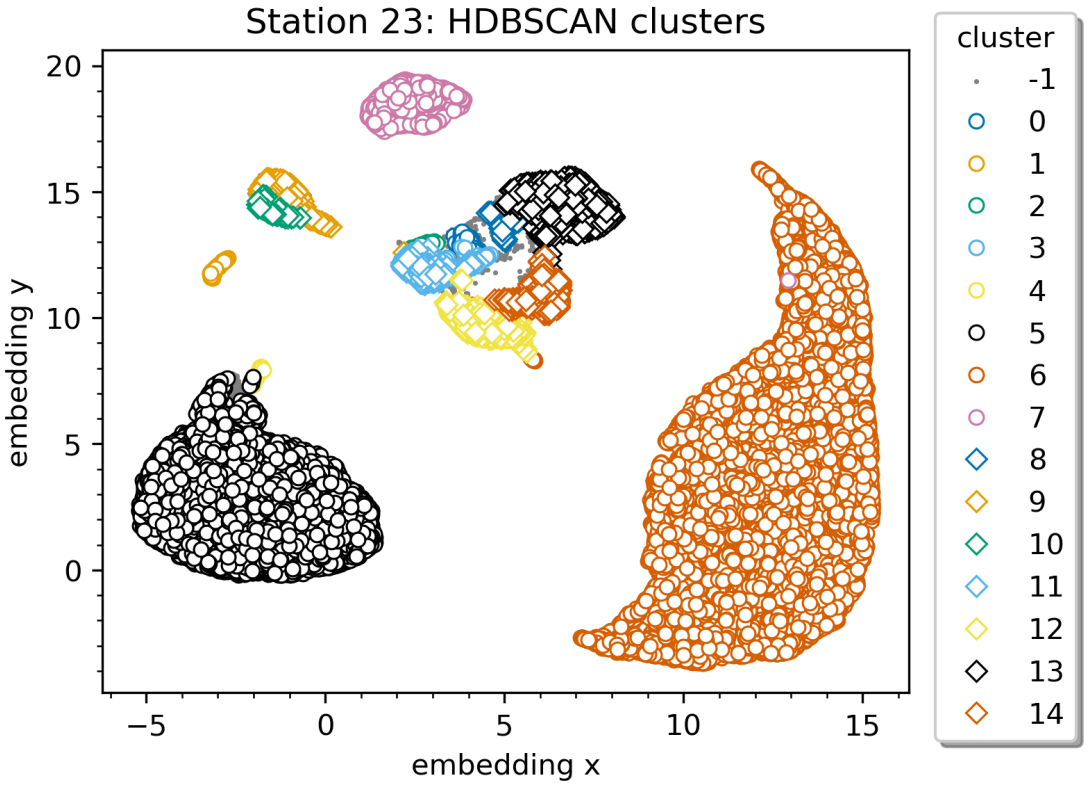


Figure 5.5.: UMAP representation of the mean latent vectors of station 23 shallow- and force-triggered events overlayed with clusters found by HDBSCAN. Cluster -1 denotes those events that are not assigned to any cluster.

considered. Here one can already make out multiple clusters by eye. Furthermore, there appear to be clusters that consist almost exclusively of events appearing at high wind speeds above 10 m/s and there are also those which contain mostly lower wind speeds. There is also a less separated group of potential clusters in the middle, which appears to contain wind speeds both a bit lower and higher than 10 m/s. Interestingly, for the high-wind cluster in the lower right of the UMAP representation, events with very high wind speeds (> 17 m/s) are positioned on the edge of the cluster. It is important to mention here that neither the variational autoencoder nor the UMAP and HDBSCAN algorithms have access to any weather or wind information.

By investigating the HDBSCAN clustering as well, one finds that the clearly separated clusters observed in the UMAP representation are indeed found to be separate clusters by HDBSCAN, although sometimes a potential cluster in the UMAP is split up by HDBSCAN. The group of clusters in the middle is separated into multiple clusters of different sizes. Almost all events in the clusters found

by HDBSCAN are also grouped together in the UMAP representation, indicating that the clusters are indeed reasonable.

Another interesting aspect of the HDBSCAN clustering is that not all events are assigned to a cluster. By comparing event IDs, it is found that two potential cosmic ray candidate events ((station 23, run 1091, ID 1718) and (station 23, run 793, ID 748)) are assigned to cluster -1 i.e. not clustered. Given the expected small number of cosmic ray events in the sub-sampled data, these events are indeed true outliers and HDBSCAN classifies them as such. However, these two candidate events may be just by chance assigned to cluster -1. This is unlikely, since the probability of coincidentally assigning both events to cluster -1 is 0.9 %, where the probability is estimated by the square of the ratio of the number of events in cluster -1 (4247) to the total number of events (45177).

5.3.4.3. Analysis of event rates and the number of events of identified clusters with respect to wind speed

To identify wind clusters, the number of events (cluster size) as a function of the wind speed is investigated per cluster. For this purpose, the *(empirical) cumulative distribution function* (CDF) of each cluster is plotted (Figure 5.6) against wind speed. One can see that there are different turn-ons for different clusters: some clusters (1, 4, 5, and 7) start to appear already at 0 m/s, while others feature a turn-on between 2 m/s and 7.5 m/s. Three clusters (6, 9, and 10) show a clear turn-on at around 10 m/s, whereby solely cluster 6 contains a considerable fraction of events at very high wind speeds of 17.5 m/s and above. The behavior of the CDF curves is for most clusters very similar to that of the CDF of a normal distribution: a continually increasing slope up to some inflection point after which the slope steadily decreases and converges to zero. However, for clusters 11 and 13, there appear three inflection points instead of one so it looks like these clusters have two turn-ons, the first between 3 m/s - 5 m/s and the second around 10 m/s. Apart from coincidence, this might come from the (unknown) nature of the underlying production process of these events, if they are indeed wind-induced. However, it is also possible that these clusters contain events of two different event classes that feature two different behaviors as a function of wind speed. To study, whether or not a single cluster is made up of multiple event classes, mean frequency spectra are investigated per cluster for different wind speed intervals in subsubsection 5.3.4.4.

One problem with the raw CDF in Figure 5.6 is that, as shown in Figure 5.7, the wind speed distribution in the investigated time interval is not flat so some

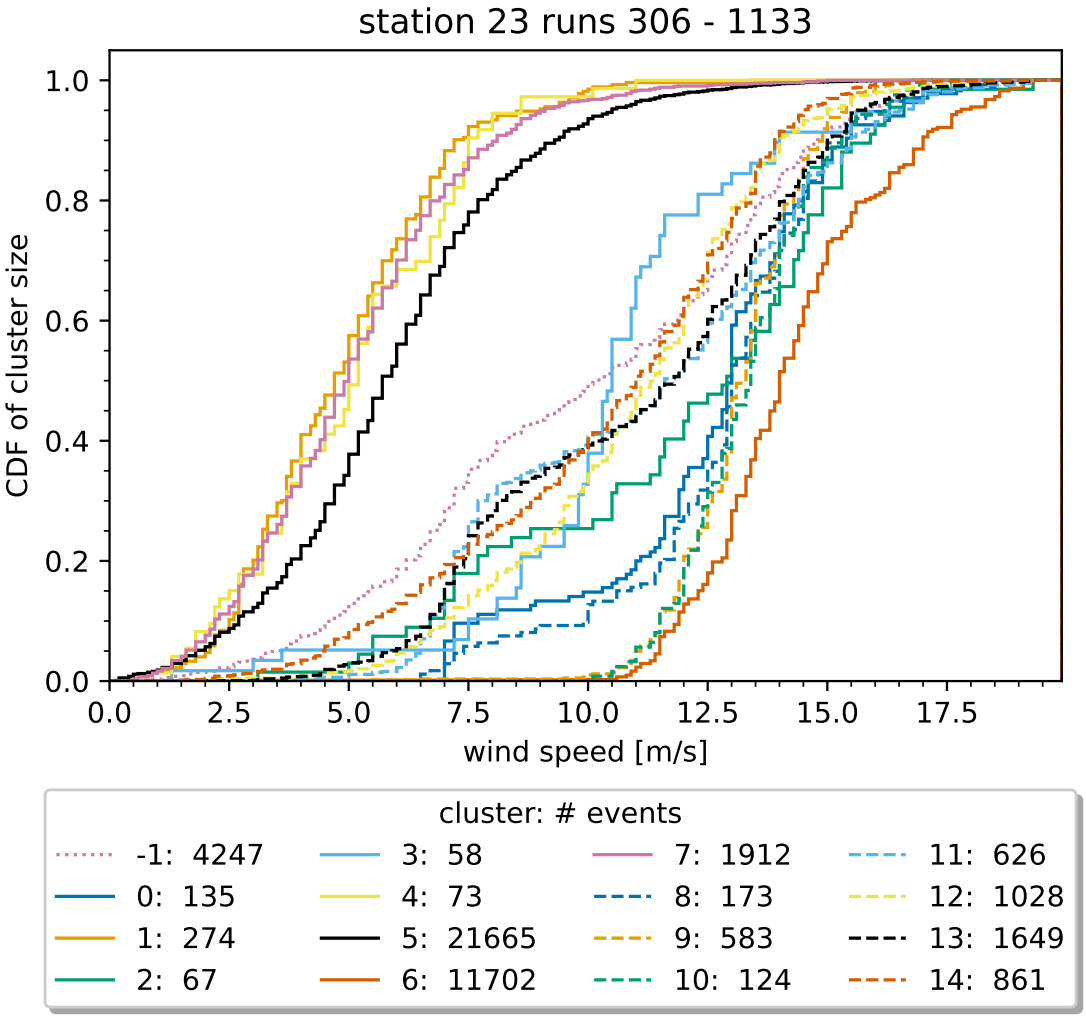


Figure 5.6.: Cumulative distribution of the number of events (cluster size) as a function of the wind speed per cluster found for station 23. Cluster -1 denotes all events not assigned to any cluster. The total number of events per cluster is shown in the legend

wind speeds are over- or underrepresented. To counter this imbalance, a *data-normalization* is introduced: for a chosen binning, a histogram with counts n_i in bin i of the weather quantity of interest (e.g. wind speed) is computed, whereby the weather data is taken every minute for the whole investigated time interval. After that, each bin count of the cluster histogram $n_{i,\,cluster}$, from which the CDF

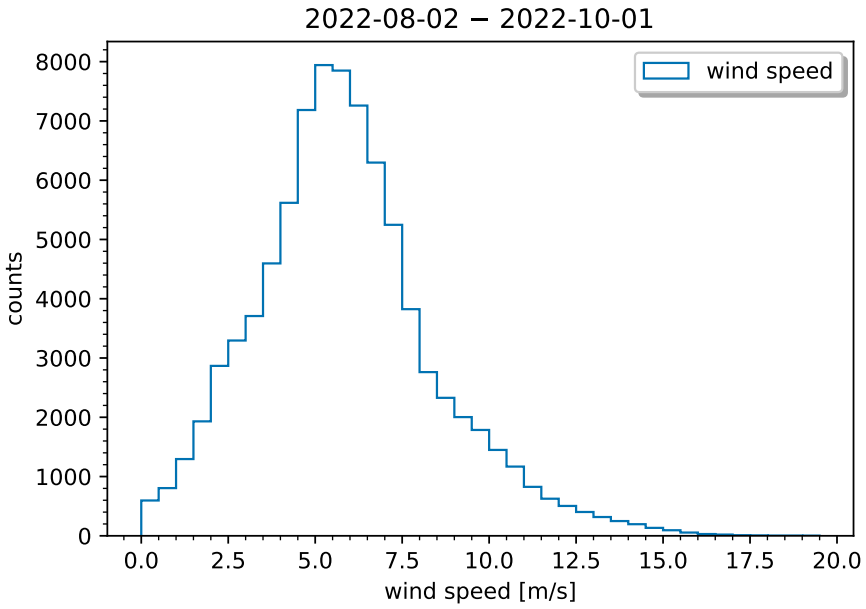


Figure 5.7.: Histogram of the wind speed taken every minute at Summit Station for the investigated time interval.

is computed, is divided by n_i to obtain the normalized cluster size per bin

$$n_{i,\,cluster}^{norm} = \begin{cases} \frac{n_{i,\,cluster}}{n_i}, & n_i \neq 0 \\ 0, & n_i = 0 \end{cases} \tag{5.2}$$

The data-normalized cluster size as a function of the weather histogram should ideally be flat/uniform for an event class independent of the investigated weather quantity. However, for a limited time interval, statistical fluctuations will lead to deviations from the flat distribution.

To better compare multiple cluster sizes as a function of e.g. wind speed, the data-normalized empirical cumulative distribution is computed. Ideally, the data-normalized CDF of a cluster that does not depend on some weather quantity should be linear with respect to said quantity, whereby the zero point of the distribution should lie at the smallest value observed for the respective quantity and the CDF should reach 1 at the greatest observed value. However, the investigated time interval of two months is rather short so some fluctuations and deviations may occur. The reason for this is that not all possible weather configurations are included and random coincidences may happen, e.g. an event class of man-made origin that does occur only during a single day coincidentally appears during a high-wind period. Additional fluctuations may also be created due to the sub-

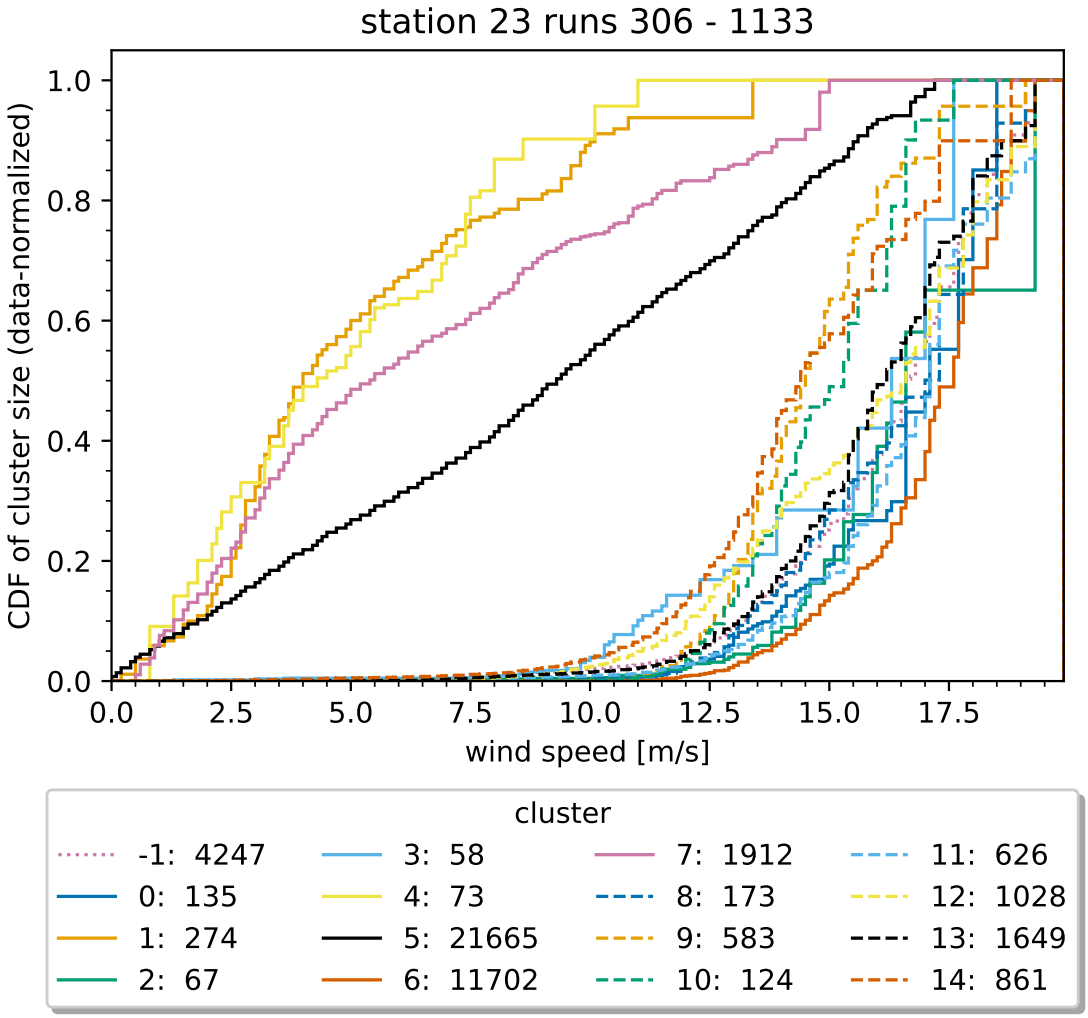


Figure 5.8.: CDF of the number of events (cluster size) as a function of the wind speed (data-normalized) per cluster found for station 23. The data-normalization takes the actual distribution of wind speed in the investigated time period into account. The total number of events per cluster is shown in the legend.

sampling of the data, since per run only about 3-5 % of all events are transferred via satellite and available for this analysis.

The data-normalized CDF of the cluster size of clusters found for station 23 as a function of wind speed is shown in Figure 5.8 with the total number of events per cluster written in the legend. Again, cluster -1 denotes all unclustered events. There are mainly three classes of clusters arising from the plot:

- Clusters that show a turn-on at wind speeds between $\sim 7\,\mathrm{m/s} - \sim 12\,\mathrm{m/s}$

and furthermore have a mostly convex curve. This is the case for clusters −1, 0, 2, 3, 6, 8, 9, 10, 11, 12, 13 and 14. The turn-on is here defined as the wind speed, where the CDF reaches 1 %.

- Clusters that have a linear behavior as a function of wind speed. As described before, such clusters are those that are truly independent of wind speed. Only cluster 5 shows this behavior. This can be explained by the trigger information, as cluster 5 contains the vast majority of force triggers which are taken every 10 seconds and are thus independent of wind speed.

- Clusters that do not have a turn-on at non-zero wind speeds and have a mostly concave curve. This can be observed for clusters 1, 4, and 7.

Interestingly, the double-turn-on behavior of clusters 11 and 13 shown in Figure 5.6 is not visible anymore. This is possibly due to the data-normalization, since the wind speeds around 5 m/s to 8 m/s are the most prominent in the investigated time interval and are therefore suppressed by the data-normalization. The first turn-on of clusters 11 and 13 lies near this region and is presumably suppressed by the data-normalization.

To further investigate clusters, it is useful to look at event rates vs. time for each cluster and compare them to wind speed. For station 23, this is shown in Figure 5.9. To display the full time span of two months, 120 minute bins are chosen and for the wind speed, the maximum over each time bin (120 minutes) is plotted. Time intervals where the maximum wind speed is greater or equal to 10 m/s are shaded in gray.

The unclustered events (cluster −1) appear almost throughout the entire time interval with increased event rates during high-wind periods. This indicates that cluster −1 contains both wind-correlated and wind-uncorrelated events. Cluster 5 (mostly force triggers i.e. thermal noise) shows an almost flat event rate for the observed time interval, whereby some of the occasional dips are caused by runs that are filtered out beforehand (calibration runs, short runs). Additional dips are created by a temporary reduction of force triggers when the shallow trigger rate is too high for a short amount of time. Apart from the dips, fluctuations in the event rate of cluster 5 can be explained by the sub-sampling of the data. Additionally, thermal noise can occasionally activate the shallow trigger, adding more statistical fluctuations. Now, looking at the remaining clusters that do not show a non-zero turn-on as a function of wind speed, there are two clusters (1 and 7) that show a periodic behavior of the event rate with time periods of several hours where the event rate is zero. By investigating the cumulative number of events as a function of the hour of the day (see Figure A.5 in appendix), one can observe that these clusters appear exclusively during daytime and are therefore responsible for the

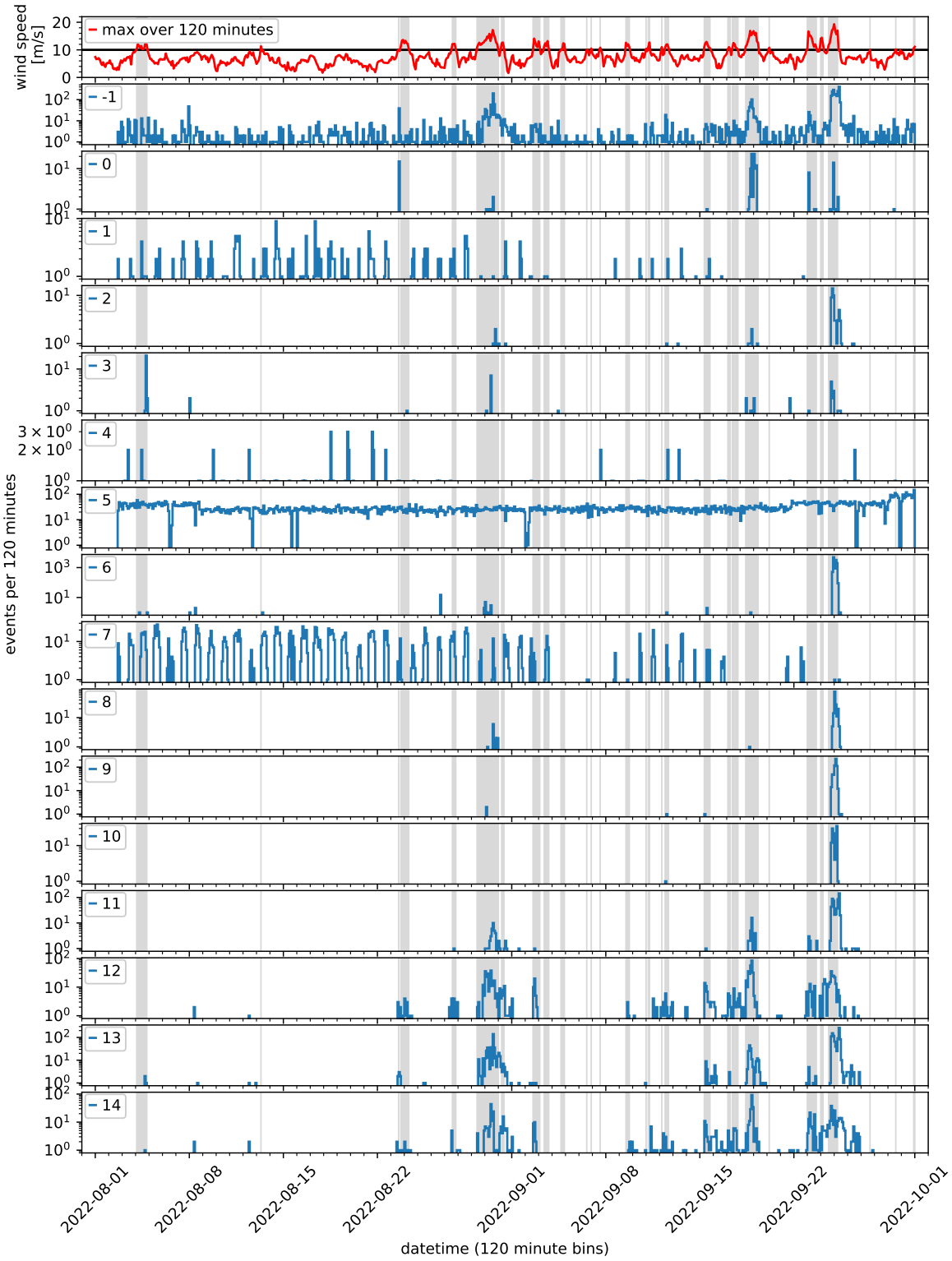


Figure 5.9.: Station 23: event rate per cluster found by HDBSCAN as well as wind speed. Time intervals with wind speed $\geq 10\,\mathrm{m/s}$ are shaded in gray. Time is given in UTC.

periodically increased event rate seen in Figure 5.1. As mentioned before, this background is unknown and may be connected to the station's solar panels or human activity. The last cluster featuring a non-zero turn-on in wind speed, cluster 4, seems to appear randomly throughout the entire time interval with some events coincidentally appearing during/near high-wind periods, indicating that cluster 4 is indeed independent of wind speed.

Among the clusters that do show a non-zero turn-on, there are some, namely clusters 6, 8, 9, and 10, where the majority of events appear during a single high-wind period of the length of roughly a day, while events of the remaining clusters are spread across multiple high-wind periods. Apart from a few events, all events of these clusters appear during or near the shaded time intervals, which is a strong indicator that these clusters indeed correlate with wind speed. The fact that some wind clusters appear almost exclusively during a single day for the investigated two months gives rise to the hypothesis that other factors besides wind speed influence the underlying processes responsible for the generation of wind-correlated events. Such potential quantities may include other weather data such as temperature, wind direction, pressure, and relative humidity. In subsubsection A.2.2.1, data-normalized cumulative distributions of the number of events per cluster as a function of these weather quantities are shown. To assess, whether some behaviors of the CDF are due to the respective weather quantity and not due to the wind speed, one must disentangle wind speed from other weather quantities. Therefore, wind speed is plotted against wind direction, temperature, barometric pressure, and relative humidity in section A.3.

For the wind direction, the wind-correlated clusters seem to appear only at certain preferred wind directions, resulting in a stair-like behavior of the CDF. Clusters 6, 9, and 10 appear exclusively in a narrow range from 300 ° to 350 °, where the exclusiveness is explained by the fact that these clusters appear only during a single day. By looking at Figure A.80a one can conclude that the stair-like behavior of the CDFs is likely caused by the wind speed being high only for certain wind directions and not by the wind direction itself. However, it can not be excluded that the wind direction influences the wind event production mechanisms so that some classes might be suppressed for certain directions. How the wind direction might influence the production of wind events is not known, but if some wind events are created at metal structures, there might be some direction, where the wind has more area of attack. This could then support or prevent the production of wind events for specific wind directions.

Figures A.80b, A.81a and A.81b show that, taking the linear weather-independent cluster 5 (thermal noise) as reference, most wind clusters appear at lower temper-

atures ($\lesssim -20\,^{\circ}\mathrm{C}$), slightly higher relative humidity ($\gtrsim 60\,\%$) and slightly lower pressures ($\lesssim 670\,\mathrm{hPa}$) while the other non-wind clusters show the opposite behavior. For the relative humidity, this is likely caused by the wind speed being high only above $R_H > 60\,\%$, while for temperature and pressure, no clear explanation is visible.

Given that the investigated time period is only two months, no definite conclusions about the influence of other weather quantities can be drawn from the data, as some weather quantities might be linked or might appear coincidentally in certain configurations during the investigated high-wind days. Therefore, more data, especially longer observation time periods are required to further analyze this phenomenon. Additionally, wind tunnel experiments, where some of the weather quantities are controllable, can help to understand wind events. Furthermore, it is possible that over a few years, more wind-correlated clusters arise, if some of the processes responsible for generating the wind-correlated signals are sensible to certain wind-weather configurations. Hence, it might be that the wind clusters found in this thesis do not fully represent the complete spectrum of wind-correlated events and it might therefore be necessary to redo the clustering analysis from time to time until the nature of these events is understood.

5.3.4.4. Investigation of mean spectra per cluster

After analyzing cluster sizes and event rates vs. wind speed, the specific event characteristics of each cluster are explored by investigating arithmetic mean frequency spectra. The mean spectra of all upward-facing channels of all clusters are depicted in subsubsection A.2.2.2 in the appendix. Additionally, waveforms and frequency spectra of randomly drawn example events are shown in subsubsection A.2.2.3 per cluster. For better readability and since the triggering signal always appears in the second half of the waveform, only the second half of the waveforms are shown. Note that these example events are solely for visualization purposes and are not necessarily representative of the cluster, although the example spectra may share some features with the mean spectra.

Starting with the mean spectra of the wind-independent clusters, it is now possible to confirm that cluster 5 indeed contains mostly thermal noise events with some narrowband peaks at $\sim 136\,\mathrm{MHz}$, $200\,\mathrm{MHz}$, $403\,\mathrm{MHz}$ and $480\,\mathrm{MHz}$, indicating that cluster 5 contains some CW events as well. Since CW events can be easily filtered out using the L1 score, this is not a problem. The peak at around $136\,\mathrm{MHz}$ presumably comes from hand-held radio devices. The origin of the spikes at $200\,\mathrm{MHz}$ and $\sim 480\,\mathrm{MHz}$ is not known and the source of the $403\,\mathrm{MHz}$ line is the weather balloon. Apart from thermal noise, the spectrum contains a small

contribution from galactic noise (more information can be found in [12]). Furthermore, it appears that the mean spectrum of channel 16 is stretched compared to the other channels, but the reason for this phenomenon is not known. Cluster 4 contains exclusively narrowband CW at $\sim 150\,\mathrm{MHz}$ and can be attributed to hand-held radio devices. The events in clusters 1 and 7 appear periodically and exclusively during daytime and both clusters have nearly identical mean spectra with two dominant sharp but not narrowband peaks at $\sim 80\,\mathrm{MHz}$ and $\sim 160\,\mathrm{MHz}$.

For the wind-correlated clusters, the mean spectra feature various peaks of different widths at different frequencies across clusters and sometimes even across individual channels within a cluster. Only clusters 9 and 10 show almost identical mean spectra. In almost all clusters, a significant fraction of the total integrated spectral power is contained in one or multiple peaks around 100 MHz. This region is amplified due to the LPDA's vector effective length, which is largest around 100 MHz. For cluster 6, the $\sim 100\,\mathrm{MHz}$ peak is very large and wide compared to most other clusters, with channels 13 and 16 being much stronger than channel 19. Other notably strong peaks are those around 200 MHz (notable in clusters (3), 12, 13, and 14, strongest in channel 16), $\sim 330\,\mathrm{MHz}$ (clusters 9 and 10, strongest in channel 16) and $\sim 500\,\mathrm{MHz}$ (clusters 9 and 10, strongest in channel 13). In all clusters except for cluster 3, channel 19 shows the weakest mean spectrum. Since the antenna sensitivity is direction-dependent, this could mean that these events, at least on average, arrive at channel 19 in a direction where the antenna is less sensitive. From this, it follows that there might be some dominant direction/solid angle window from which these events originate since one would expect similar mean spectra in all channels if the directions were uniformly distributed. These events may be created at or near man-made structures as suggested in [7]. However, a detailed direction reconstruction is necessary to obtain more insights into the origins of each cluster.

As mentioned earlier, a double-turn-on is observable in Figure 5.6 for clusters 11 and 13. A possible explanation is that these clusters consist of two or more sub-clusters, each having its own turn-on. To investigate this possibility and to further assess the purity of the found clusters, mean frequency spectra of certain wind speed intervals are compared per cluster and channel, whereby the plots can be found in subsubsection A.2.2.4. The investigated wind speed intervals are $n \cdot 5\,\mathrm{m/s} \leq v_{wind} < (n+1) \cdot 5\,\mathrm{m/s}$ for $n = 0, 1, 2, 3$ (all wind speeds measured are below $20\,\mathrm{m/s}$) and spectra for intervals containing no events are set to zero. For all clusters, except for -1, 0, 6, and 9, the mean spectra of all wind speed intervals with at least one event closely match each other within statistical fluctuations, indicating that each of the clusters may indeed contain only a single event class. Sometimes only a few events are present in an interval resulting in noise not being completely averaged out. It is however still possible that a clus-

ter contains multiple event classes both with similar appearance per wind speed bin. The mean spectra of clusters 11 and 13 are almost identical across different wind speed intervals, hence the double-turn-on behavior can not be explained by potential sub-clusters with individual turn-ons. For cluster 9, only four events are contained in the $[5, 10)$ m/s interval, but some peaks of the full mean spectrum are already visible but far weaker than in the higher wind speed intervals. A similar observation can be made for cluster 0, where the mean spectra in the $[5, 10)$ m/s interval do not show all features of the two higher wind speed intervals, although the peak at ~ 100 MHz is very similar in all intervals. For cluster 6, the strength of the ~ 100 MHz peak increases with increasing wind speed. This might indicate, that there is a connection between signal strength and wind speed. Cluster -1 contains all unclustered events and therefore it is expected that the mean spectra of different wind speed intervals do not match. For the two intervals with the highest wind speeds, however, the mean spectra of cluster -1 are very similar. This means that for wind speeds greater than 10 m/s, most events in cluster -1 correlate with wind speed. It is likely that these events correspond to the various wind clusters, but show too weak of a similarity to be assigned to these clusters.

5.3.4.5. Analysis of waveform amplitudes per cluster

As observed before, the mean spectrum amplitude of the ~ 100 MHz peak of cluster 6 increases when going to higher wind speeds. One possible explanation is that for higher wind speeds, the signal amplitude in the waveform is higher, resulting in a higher signal peak amplitude in the spectrum, while the contributions to the spectrum from thermal noise are unchanged. This would also match the observation reported in [7] and mentioned in section 3.4 that some experiments found wind-correlated signals to be stronger at higher wind speeds. To study these findings, in subsubsection A.2.2.5 wind speed is plotted as a function of the maximum of the absolute waveform in a 2D histogram for each cluster and channel.

Indeed, for cluster 6 a trend of the maximum amplitude towards higher wind speeds is clearly visible, in particular in channel 16. For some other clusters (2, 3, 8, 11, 12, 13), there are slightly more high-amplitude outlier events at higher wind speeds above 10 m/s. However, the majority of events of wind-correlated clusters appear above 10 m/s, so this could be due to statistics. In contrast, cluster 9 shows a broad band in the plot and the amplitude distribution is similar across wind speeds, not counting wind speeds with only a few events. This indicates that events in cluster 9 are neither stronger nor weaker at higher or lower wind speeds. It is furthermore observable that for many wind-correlated clusters, the events at the lowest wind speeds occurring in the respective cluster have low maximum amplitudes. However, it should again be pointed out that the wind speed is always

a minute average at a location a few kilometers away from station 23, so the true wind speed at the exact time and (unknown) location of the event production may deviate from the values presented here. Therefore, the wind speed of outlier events in these plots should be treated with caution. For clusters 11 and 13, those that show the double-turn-on behavior in Figure 5.6, one can observe that there are two peaks in the wind speed vs. maximum amplitude distribution, one below and one above 10 m/s, both featuring similar numbers of high-amplitude outlier events. It looks like there are two overlapping distributions, however since the corresponding mean spectra do not change much as a function of wind speed, both distributions are expected to originate from the same physical process or at least from similar ones. The wind-uncorrelated clusters show no dependency of the maximum waveform amplitude on the wind speed, as one would expect.

In the section before, it is observed that some clusters have one channel, where the spectrum amplitude is much smaller than the remaining two. This can also be seen in the maximum of the waveform amplitudes, e.g. channel 19 for clusters 6, 12, and 14. This indicates a preferred arrival direction of signals of such clusters.

After analyzing all clusters of station 23, clusters 0, 2, 3, 6, 8, 9, 10, 11, 12, 13, and 14 are identified as wind clusters and are therefore selected for the training of the wind event discrimination in chapter 6.

5.3.4.6. Comparison of the clustering results to existing work

Before proceeding to the clustering results for other stations, the clustering found in this thesis for station 23 is briefly compared to the clustering found in [39] for RNO-G station 23. The clustering methods ([39] and this thesis: VAE and HDBSCAN) and the investigated time intervals ([39]: July to October of 2022; this thesis: August to October of 2022) are similar, so a comparison is possible. In [39], three clusters were identified in total, where two are correlated with high wind speed and the remaining cluster was attributed to thermal noise. In contrast, eleven wind clusters are identified in this thesis, where two of them (9 and 10) are very similar. Furthermore, a thermal noise cluster, a CW cluster, and two very similar non-wind background clusters of unknown origin, which appear periodically and exclusively during daytime, are found in this thesis. By analyzing mean frequency spectra and event rates per cluster, one can conclude that most of the wind clusters found in this thesis for station 23 represent separate wind event classes. This indicates that the clustering found in this thesis is more fine-grained than the clustering found in [39].

5.3.5. Clustering results for stations 13 and 24

In addition to station 23, event clustering is done for stations 13 and 24 as well. Both stations were deployed in 2022 and should not suffer from wind turbine, battery charging, and LoRaWAN noise, just like station 23. The only major backgrounds expected to appear in stations 13 and 24 are thermal noise, CW, and wind, hence wind should make up almost all impulsive shallow-triggered events, apart from a few cosmic ray events. In contrast to station 23, stations 13 and 24 feature a higher trigger threshold and therefore fewer events are expected to appear compared to station 23. As for station 23, shallow (radiant) trigger rates as well as the mean wind speed is plotted for stations 13 and 24 in Figure A.3 in the appendix. For the three major shaded time intervals where the mean wind speed was over 10 m/s an increase in event rates is observable for both stations 13 and 24, however not as high as for station 23, possibly due to the trigger threshold. Compared to station 23, stations 13 and 24 do not show a periodic increase in event rates, hence this background is not present or suppressed in stations 13 and 24, since these events have relatively small amplitudes and are not able to trigger in significant numbers.

For both station 13 and 24, the clustering procedure is the same as for station 23. The bottleneck size is again 9, the weight for the KL loss is now $\beta = 1$ for both stations and parameters used for the HDBSCAN clustering algorithm are again individually adjusted for each station. The final non-default HDBSCAN parameters used for stations 13 and 24 can be found in Table A.4 and Table A.5. The results, including UMAP representations, CDFs of cluster sizes, mean spectra, and event rates per cluster, are shown in subsection A.2.3 and subsection A.2.4 in the appendix. For both stations, there is a large thermal noise cluster (station 13: cluster 7; station 24: cluster 0) and several smaller clusters that feature a turn-on at around 10 m/s (station 13: clusters 0, 1, 2, 3, 4; station 24: clusters 1, 2, 3, 4). By looking at the event rates per cluster, all those wind clusters appear almost exclusively during high-wind periods. Additionally, for station 13 two CW clusters (5 and 6) are identified, both having narrowband peaks at 403 MHz i.e. the weather balloon. Comparing wind clusters of stations 13 and 24 to those found for station 23, one can observe that there are some similarities between clusters: many clusters show a large frequency content around 100 MHz, some exclusively, some in combination with a considerable frequency content around 200 MHz or 500 MHz. Sometimes the dominant channels are different across stations, which could mean that the arrival direction varies from station to station if those signals stem from specific structures. Looking at the clusters that feature a peak at around 500 MHz (station 13: clusters 3 and 4, station 23: clusters 9 and 10, station 24: cluster 4), one can see that the spectra deviate across stations. This may be either because those clusters might not all stem from the same process, or there are minor

differences across stations that may affect the spectra. Such potential differences include varying positions of man-made structures (e.g. solar panels) with respect to the shallow antennas. This could lead to different signal arrival directions, which might affect the spectra. Moreover, some wind clusters of stations 13 and 24 contain only a few events, so deviations due to low statistics are possible as well.

6. Wind event discrimination

In the last chapter, wind event clusters of stations 13, 23, and 24 were identified and analyzed. Now, the goal is to develop a method capable of discriminating between wind events and non-wind events. This is discussed in the following chapter.

6.1. Strategy

The task can be interpreted as a binary classification problem, where class A are all wind events and class B are all non-wind events. One could then train a binary neural network classifier on both classes. However, not necessarily all possible wind and non-wind event classes are known. A binary classifier trained on the available event classes might learn to discriminate between both classes by certain properties that are different between the known subset of classes but would have problems on unknown classes. Consider the scenario where for the non-wind events, only thermal noise is known. In contrast to wind events, thermal noise is not impulsive, making it relatively easy to discriminate between wind events and thermal noise. However, it might then be difficult for the classifier to classify impulsive non-wind events, since, for the classifier, they may look more similar to wind than to thermal noise. Additionally, if one trains a classifier using simulated events (e.g. cosmic rays) that contain some simulation artifacts, the classifier might have difficulties recognizing real events without the artifacts.

An alternative approach is to use the anomaly detection technique introduced in subsection 4.9.3. The idea is to train an autoencoder model exclusively on the previously selected wind event classes. Since the model never learns to encode and decode non-wind events, non-wind events should ideally show anomalously high reconstruction errors compared to the wind event classes the model was trained on. Discriminating wind events from non-wind events via this method requires selecting a threshold on the reconstruction loss, where events falling below this threshold are classified as wind. The value of the threshold is often chosen according to some target, e.g. such that a high fraction of wind events is correctly classified as wind while only a small fraction of (certain) non-wind events is falsely classified as wind. Although this effectively results in a binary classifier that depends on the knowledge about non-wind events, the threshold can be chosen and adjusted at all times for individual analyses. Thus, this method is ideally more robust with respect to unknown or very rare non-wind event classes, where no or

too little data is available at the time of training.

Therefore, the anomaly detection method is chosen in this thesis to discriminate wind from non-wind events. A variational autoencoder is used, where the reconstruction loss is calculated from the mean latent space vector z_{mean}. The mean latent space vector is chosen over the sampled latent variable z to obtain deterministic results.

6.2. Selection and preparation of training data

Although there are some similarities between wind clusters of different stations, they are by no means identical. Furthermore, there is a significant difference in the number of events between stations. This is at least partially due to different trigger configurations resulting in station 23 having an order of magnitude more events than stations 13 and 24. Due to the differences in wind event clusters between the stations, a variational autoencoder is exclusively trained on data of a single station. Here, station 23 is picked, since it shows more event classes and also more events in total than the other two stations. Wind events of other stations are investigated with respect to the reconstruction loss to test if the variational autoencoder can be used on other stations as well.

However, in the last chapter, it has been observed that the signals of some clusters may have a preferred arrival direction. If these signals indeed originate from man-made structures at the station, the arrival direction may vary across stations if these structures are positioned differently from station to station. It is also possible that this preferred direction depends on other factors and may change over time. Apart from varying relative frequency contents in different channels, such a preferred direction results in a preferred time difference between pulses across different channels. When training the VAE on all channels at the same time per training sample, the VAE might develop a bias towards these preferred directions and the specific timing configuration. Shifting the waveforms in time to eliminate this bias is disfavored, since, as mentioned before, the waveforms are often clipped. Thus, to make the VAE more robust against bias from channel configurations, the waveforms of each channel are treated as individual training samples. Apart from that, the data is prepared in the same way as described in subsection 5.3.2, i.e. the squared of the STFT magnitude is computed per waveform, then downsampled to 128×128 pixels and normalized to $[0, 1]$. Additionally to a two out of three channels SNR≥ 5 cut, a three out of three channels L1≤ 0.6 cut is applied to filter out any CW signals from the wind event classes.

As training data, all wind event classes of station 23 are selected. Since the VAE is evaluated later, a test data set is required apart from the training and

validation sets. This is necessary since the VAE is selected based on the validation set, which results in a bias towards the validation set. Here, 60 % of the data is used for training and 20 % is used for validation and testing, respectively. However, the clusters have strongly varying numbers of events, which would result in a potential bias towards the bigger clusters. To counter this, a method known as *undersampling* is used, where the maximum number of samples per cluster used for training and validation is limited and all remaining samples are used for testing. Here, the limit for the training data is chosen to be 2000 and the limit for the validation data is then given according to the proportions of training and validation data. The number of unique events in the training, validation, and test sets is then

$$N_{train,\,unique} = 4983, \quad N_{val,\,unique} = 1663, \quad N_{test,\,unique} = 7938 \tag{6.1}$$

Furthermore, as there are still very small clusters, samples of smaller clusters are reused n times inside the training and validation sets until either the limits used for undersampling are reached or n reaches a chosen limit (here 5). This is called *oversampling*. The limit on n is here chosen to be small to reduce overfitting. The combination of over- and undersampling reduces over- and underrepresentation of certain wind event classes, but can certainly not fully compensate lack of training data. After applying SNR and L1 cuts as well as over- and undersampling, the following number of events are available for training, validation and testing:

$$N_{train} = 12539, \quad N_{val} = 4193, \quad N_{test} = 7938 \tag{6.2}$$

Since each channel is treated as a separate training/validation sample, the number of training and validation samples is $3N_{train}$ and $3N_{val}$.

The waveforms used for training were recorded with a sampling rate of 3.2 GHz. However, it is planned to reduce the sampling rate to 2.4 GHz in the future to allow longer waveforms. Ideally, the waveforms should then no longer be clipped at the end. Therefore, the whole procedure of data preparation should be redone once the sampling rate has changed and a new VAE should be trained to be used with the new format.

6.3. Model architecture

The chosen model architecture (see subsection A.4.1) is the same as described in subsection 5.3.3, except for some minor changes. The bottleneck size is reduced from 9 to 6 since the number of channels per individual sample is reduced. Furthermore, the input is now of shape $128 \times 128 \times 1$ instead of $128 \times 128 \times 3$. The dropout rate is increased to 0.7 to increase regularization and reduce overfitting

because fewer training data is available and some samples appear multiple times due to the oversampling. Additionally, four instead of five convolutional clocks are used.

6.4. Training

As the reconstruction loss function, the mean squared error (MSE) is chosen, since this loss function suppresses thermal noise compared to the main signal. The weight on the KL loss is now $\beta = 0.1$. It is chosen to be small because the focus now lies on the reconstruction capabilities of the model and a regularization of the latent space is not of primary importance. Again, the Adam optimizer is chosen with a learning rate of 4×10^{-4}. A batch size of 32 is used as well as model checkpointing and early stopping. In the end, the model with the lowest validation MSE loss is used for further analysis. Both total validation and training loss are shown in Figure 6.1. Validation and training loss decrease at first until the validation loss reaches its minimum after which the validation loss slightly increases, whereas the training loss still decreases. Additionally, the training loss is much smaller than the validation loss. This is a strong indicator of overfitting during training. It is expected that this problem will be reduced by increasing the number of training and validation samples.

6.5. Results

In this section, the discriminator trained on wind events of station 23 is tested with respect to clusters of station 23, cosmic ray simulations, and other stations.

6.5.1. Choice of evaluation loss

Before evaluating the VAE on station 23 clusters, cosmic ray simulations, and other stations, one must choose how to calculate the evaluation loss. As mentioned above, the loss function is chosen to be the mean squared error. However, as there are three upward-facing channels per event, there are also three loss values per event (one per channel). There are several ways to compute a single loss value out of these three channel loss values, e.g. by taking the maximum, arithmetic mean, or minimum. Here it is useful to remember that, depending on the arrival direction, a signal can hit a channel in a sensitive or insensitive region of the LPDA. If a signal arrives at an insensitive region, the amplitude of the waveform in that channel is suppressed and may even fall into the thermal noise regime. Hence, it is not desirable to use a channel, where the signal is near thermal noise, because the contribution of thermal noise to the reconstruction loss is high. This comes from

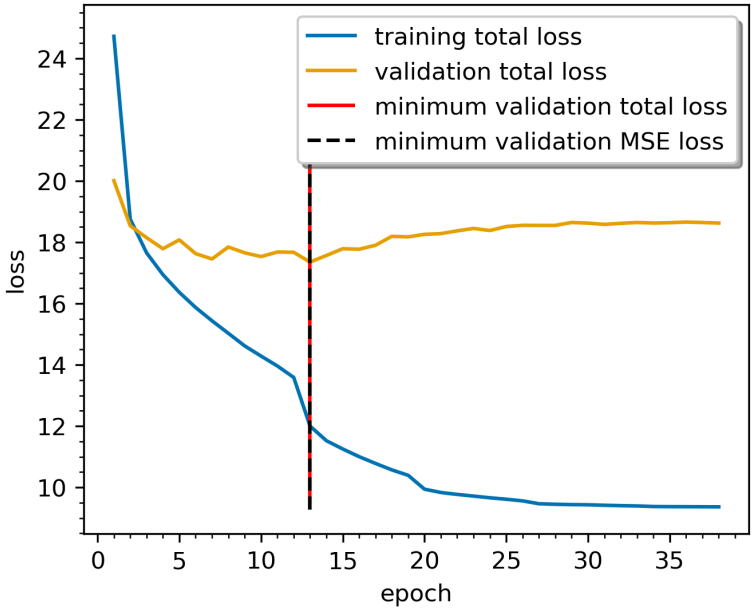


Figure 6.1.: Total validation and training loss as a function of the training epoch for the VAE used for the discrimination of wind from non-wind events. The reconstruction loss is here the mean squared error (MSE). Marked are both the minimum total validation loss and the minimum validation MSE loss.

the fact that the input data is normalized to $[0, 1]$, so if the signal is much stronger than thermal noise, thermal noise is suppressed in the mean squared error. As a result, the maximum and mean loss across channels are not used to calculate the final per-event loss, as both potentially include a channel, where a weak or even no signal is present, increasing the mean/maximum loss. The remaining choices include the minimum loss across channels and the mean of the two lowest loss values of all three channels. The latter is motivated by the fact that at least two channels are required to have a signal above the trigger threshold to trigger an event. However, the trigger threshold is variable and may be near the thermal noise regime. Thus, it is possible that for some events, depending on their arrival direction, there is only a single channel that shows a clear signal. Therefore, the minimum loss of all three upward-facing channels is chosen as evaluation loss for further analysis.

6.5.2. Evaluation of station 23 events

Now, events of station 23 are evaluated per cluster. Since the VAE is trained on wind clusters of station 23, only the test data per wind cluster is considered. The empirical CDF of all shallow-triggered events of all clusters as a function of the

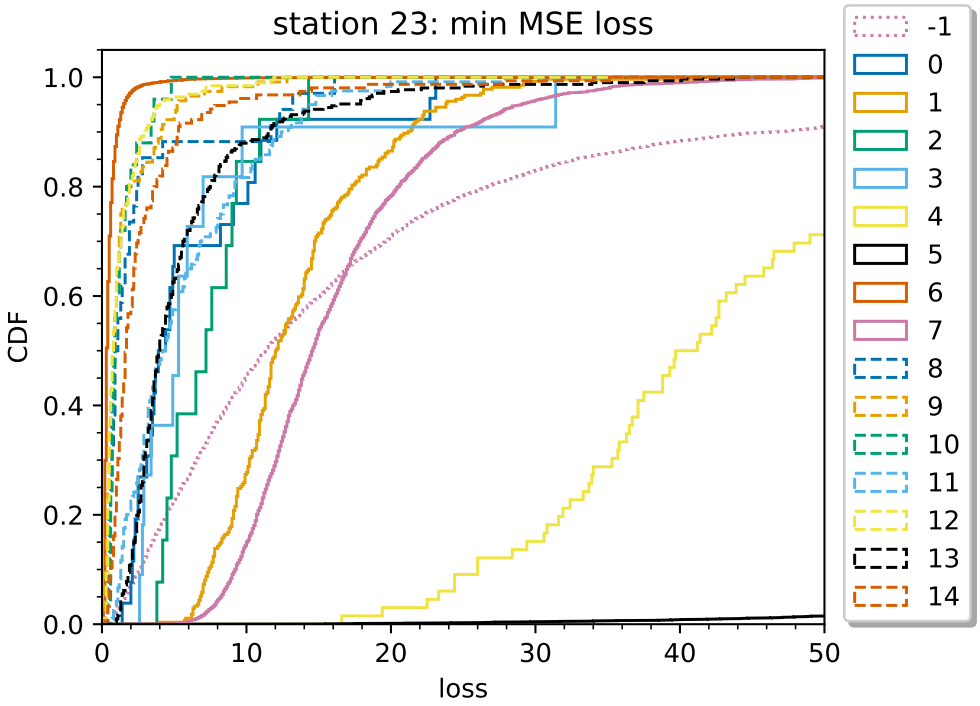


Figure 6.2.: Station 23: cumulative distribution of the minimum mean squared error (MSE) across all three channels per event cluster. Only shallow-triggered events are included. Clusters 1, 4, 5, and 7 are identified as non-wind clusters. Cluster −1 contains all unclustered events, which consist of both wind and non-wind events. Cluster 5 is mostly thermal noise and has only a few events in the shown loss interval.

minimum MSE loss is shown in Figure 6.2. Overall the wind clusters (0, 2, 3, 6, 8, 9, 10, 11, 12, 13, 14) show a lower loss than the non-wind clusters (1, 4, 5, 7). The CDFs of some clusters are very jagged because they contain only a few events in the test data set. For cluster 5, the thermal noise cluster, only a small fraction of all events are contained in the shown loss interval. Among the wind clusters, two groups are emerging in the CDF: on the one hand clusters 6, 8, 9, 10, 12, and 14 with the lowest loss and on the other hand clusters 0, 2, 3, 11, and 13 with a slightly higher loss. Although the two day-night variation clusters, 1 and 7, appear at a higher loss than the wind clusters, there is considerable overlap with some wind clusters. The CW cluster (4) in contrast, is more separated from the wind clusters since wind events are impulsive and CW events are usually not. Cluster −1, the unclustered events, contains both wind and non-wind events and thus some cluster −1 events appear at lower loss values. However, cluster −1 may contain wind events with a high loss, which do not share similarities with other

wind clusters.

All in all, these results deviate from the ideal separation of wind and non-wind clusters without any overlap. In particular clusters 0, 2, 3, 11, and 13 show a considerable overlap with non-wind clusters. As clusters 0, 2, and 3 contain only a few events, separation is expected to improve if more data is available for training. Furthermore, some wind clusters may contain a few non-wind events and vice versa, resulting in more overlap between wind and non-wind clusters.

Apart from the overlap, it is in principle possible to discriminate wind from non-wind events, whereby the overlap varies from cluster to cluster and is expected to decrease if more training data is available. Due to the overlap, a low threshold of $L = 5$ on the loss is required, if e.g. a very pure set of wind events without non-wind backgrounds is desired for analysis.

An illustration of how events with high (> 10) and low (< 2) loss may look is shown in subsection A.4.2 for example events of the test data set. The left part of the plots shows the original spectrogram of the waveform of an event that is then given as input to the discriminator. The output of the discriminator is shown in the middle of the plots and the absolute of the difference between input and output is shown on the right.

In Figure A.82, one can see that the reconstruction for the example event with high loss still shows the characteristics of the input. However, multiple small deviations between input and output, including contributions from thermal noise due to the lower SNR compared to the example event with lower loss, still result in a high loss.

For cluster 12, the plots in Figure A.83 show a low loss example event with very accurate reconstruction and a high loss event, where the reconstruction is not successful. Interestingly, the high loss event appears to be of very similar shape until $\sim 590\,\mathrm{ns}$, where a second signal appears, which is cut off at the end. This might be a rare variation of a cluster 12 event or even a rare superposition of two signals, which the discriminator is apparently not familiar with. Another example of a potential variation of an event within a cluster is depicted in Figure A.85 for cluster 14. Here, both the low and the high loss event show a blob at $\sim 200\,\mathrm{MHz}$ followed by a weaker blob at $\sim 100\,\mathrm{MHz}$. However, for the high loss event, there appears to be an additional structure around the $\sim 200\,\mathrm{MHz}$ blob, resulting in a high loss.

Apart from smaller variations and contributions by thermal noise, there are also cases, where there are huge differences between events inside a cluster. An example of this is shown in Figure A.84, where the high-loss event shows a much more complex structure than the low-loss event. Whether both events are of the same class or one of them is an impurity of the cluster is not known.

Although these example events do not necessarily represent their respective cluster, they indicate that small variations in the spectrogram can cause a huge increase in reconstruction loss. Therefore, more data is necessary to cover more of these potential variations.

6.5.3. Evaluation of cosmic ray simulations

As mentioned earlier, the upward-facing LPDAs can indirectly measure cosmic rays via radio emission of the particle shower created by the interaction with the atmosphere. For an optimistic scenario, approximately ~ 1 cosmic rays per station per day [9, 42] are expected, depending on the trigger. Hence, finding cosmic ray signals among the many backgrounds in the data is a difficult task. One way to search for cosmic rays in the bulk of available data is to prepare a cosmic ray template waveform from simulations and then perform a template search by analyzing the correlation of event waveforms with the template [43]. If the simulations are realistic enough, real cosmic ray events should show a higher correlation score compared to background events. However, impulsive signals such as wind events sometimes reach a relatively high correlation score as well, complicating the cosmic ray search. For such a case, a wind event discriminator can be employed to apply an analysis cut to reduce interference of wind events with the analysis. Without such a wind discriminator, one would have to reject all time intervals with high wind speed, reducing observation time. Furthermore, event rates sometimes stay high for some time even after the wind speed decreases to lower levels ("afterglow" effect observed in section 5.2), complicating such a wind speed cut.

To create an analysis cut using the wind event discriminator, one must choose a loss threshold depending on the desired behavior of the cut. For some cosmic ray analyses, one may choose a threshold, where only a small fraction, e.g. 1 %, of all cosmic rays are falsely classified as wind events (false positives). On the other hand, for some analyses, the fraction of wind events correctly classified as wind (true positives) should be high, e.g. 80 %, for an analysis cut to provide a significant benefit. As the classifier is not perfect, there is always a trade-off between true and false positives and hence there is no universally correct loss threshold.

To pick a threshold, the discriminator must first be tested on cosmic ray events. However, cosmic rays in the energy range targeted by RNO-G are rare and only a few candidate events are found. Therefore, cosmic ray air shower simulations are used instead. These simulations are created for the RNO-G Collaboration using CORSIKA [44] for the shower simulation and CoREAS [45] for the simulation of radio emission, whereby energy and zenith angle of incoming cosmic rays are varied. The zenith angle ranges from 5 to 85 in steps of 0.5 and the energy ranges from 10^{16} eV to $10^{18.5}$ eV, where the exponent is increased in steps of 0.5. Since

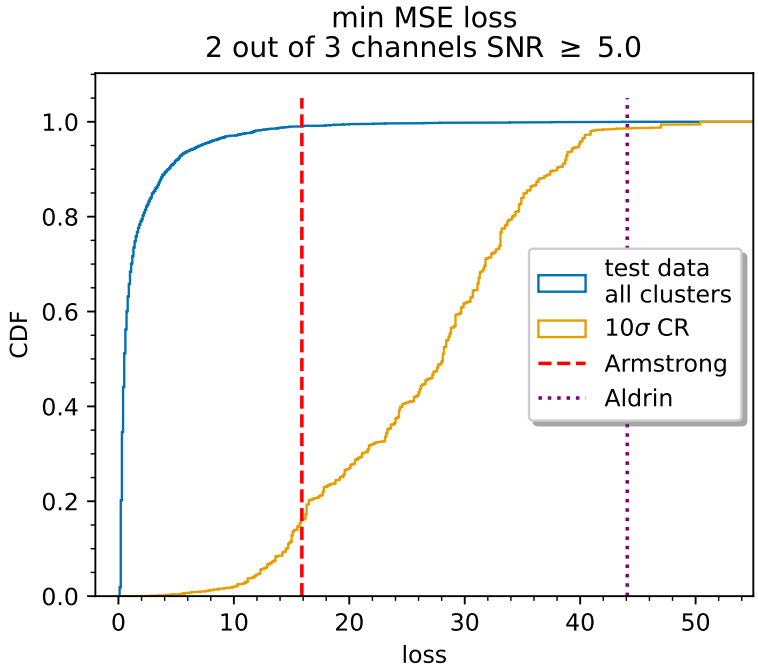


Figure 6.3.: Cumulative distributions of the minimum loss of full test data (from all wind clusters) of station 23 and cosmic ray simulations for a 10σ trigger. A two out of three channels SNR $\geq$ 5 cut is applied to the data. Two cosmic ray candidate events (Armstrong and Aldrin) are marked by vertical lines. The test data is weighted according to the appearance of clusters in station 23 and the cosmic ray simulations are weighted according to the expected cosmic ray flux per day. The loss is calculated by taking the minimum MSE across all three channels.

Earth's magnetic field is almost vertical (≈ 0 zenith) at Summit Station, the impact of the azimuth angle is assumed to be negligible [42]. Therefore, the azimuth is kept constant for all simulations. For the cosmic ray waveform simulations, a detector array of all 35 stations is simulated. Each simulated station is modeled according to a real RNO-G station, including antenna and hardware response. Per energy-zenith configuration, the simulation is repeated multiple times, whereby the detector array is rotated randomly to cover multiple signal arrival directions and antenna sensitivities. A two out of three channel envelope coincidence trigger is used for the simulations with a trigger threshold of 10σ, where σ denotes the RMS of simulated noise after a 80 MHz − 180 MHz bandpass filter. In total, 1858 triggered cosmic ray simulation events are available for this analysis.

In Figure 6.3, CDFs of the minimum loss of cosmic ray simulations and combined station 23 test data of all wind clusters are shown. The loss of two cosmic ray

candidate events is shown as well, where Armstrong denotes (station, run, ID) = (23, 1091, 1718) and Aldrin denotes (23, 793, 748). Since a two out of three SNR$\geq$ 5 cut is applied to the test data, the same cut is applied to the cosmic ray simulation waveforms.

For the raw cosmic ray simulation data sets, the energies and zenith angles are not represented as expected from their physically expected distributions. Hence, some energies and zenith angles are over- or underrepresented in the data set. Therefore, a weighting factor is computed per simulation sample according to their energy and zenith angle and how often each configuration triggers. For this purpose, the predicted cosmic ray rate per day per energy and zenith bin is calculated by Jakob Henrichs from the cosmic ray flux and the detector's simulated effective area for a specific trigger threshold. Here, the Auger 2019 cosmic ray flux [46] is used and the effective area is simulated by Lilly Pyras. Then, the number of triggered simulations per energy-zenith bin is determined, whereby a simulation configuration that does not fall into any bin is assigned a weighting factor of zero. The final weighting factor per energy-zenith bin is given by the expected cosmic ray rate divided by the number of triggered events in that bin. The code used to calculate the weighting is adapted from code by Jakob Henrichs.

The combined test data includes the individual test data sets of all clusters. However, due to the undersampling method, the test data sets do not represent the true occurrence of clusters in station 23 during the investigated time period of 2 months. Therefore, each event is assigned a weight, such that the relative cluster sizes within the test data set are representative of the relative cluster sizes within the full data set, which is composed of training, validation, and test data.

As can be seen in the plot, test data and cosmic ray simulations are separated, so that a high fraction of test data can be filtered out without touching the cosmic ray simulations. The candidate events are positioned on the ends of the loss distribution of the cosmic ray simulations with Armstrong being in the lowest 20 % and Aldrin in the highest 5 % of the loss distribution.

In Table 6.1, various loss thresholds L and corresponding fractions of test data and cosmic ray simulations with loss $\leq L$ are shown. Events with loss $\leq L$ are classified as wind. One can observe that the CDF of the test data initially rises very fast and allows high fractions of test data to be correctly classified as wind, while only small fractions of the cosmic ray simulations are falsely classified as wind. However, further increasing the threshold to increase the fraction of test data from e.g. 95 % to 99 % increases the fraction of falsely classified 10σ cosmic ray simulations from 0.906 % to 13.9 %. Thus, for a cosmic ray analysis, this suggests that there is almost no drawback in cutting 95 % of the test data, but the last few percent can cost a high fraction of potential cosmic ray events. Looking back at

loss threshold L	% test data $\leq L$	% 10σ CR $\leq L$
2.16	80.1	0.008
4.37	90.0	0.265
7.15	95.0	0.906
7.48	95.3	1.00
15.3	99.0	13.9

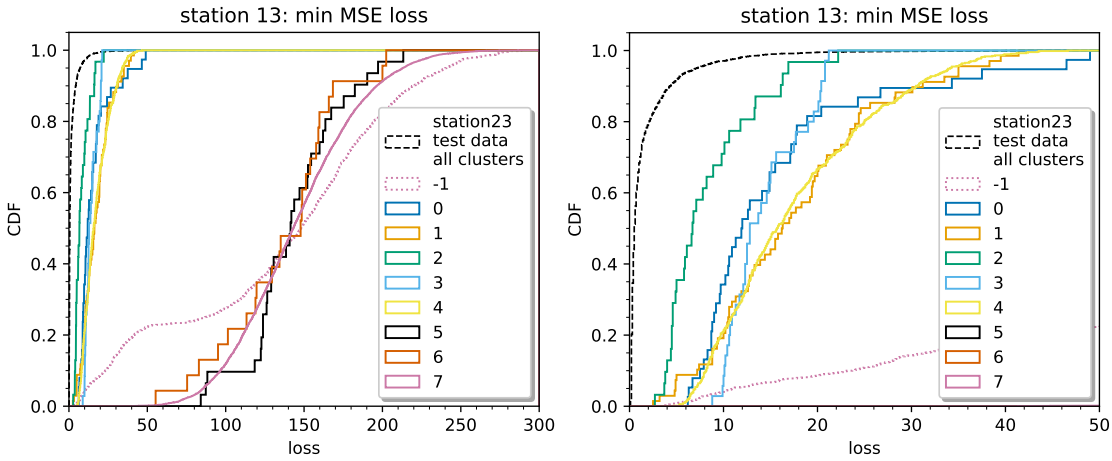
Table 6.1.: Thresholds on the MSE loss and corresponding fractions (in percent) of test data and cosmic ray (CR) simulations that are below or equal to the threshold i.e. classified as wind. The simulations are produced for a 10σ trigger.

the loss of individual clusters (see Figure 6.2), one can conclude that this behavior is caused by individual wind clusters that feature a higher loss than others. The separation between test data and cosmic ray simulations is expected to improve by using more training data. However, it should be pointed out that the test data does not necessarily represent all wind events, since cluster -1, which contains wind and non-wind events, is not used for training, validation, and testing. Hence, there may be wind events in cluster -1 with a high loss, which would decrease the fraction of wind events equal or below a loss threshold. Thus, reducing the number of unclustered wind events is important for future clusterings.

6.5.4. Investigation of other stations

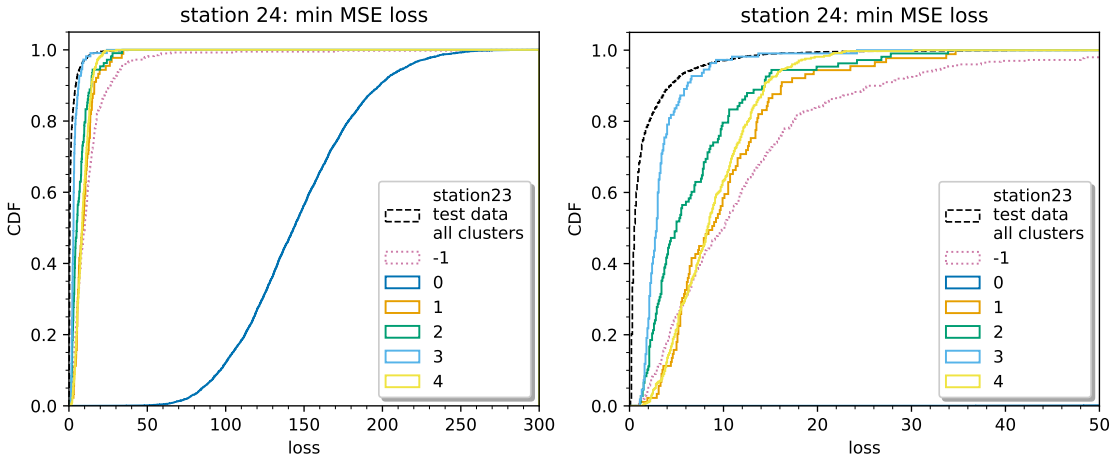
Now, the discriminator is tested on events of other stations. Firstly, station 23 test data is compared to the event clusters found for stations 13 and 24 in Figure 6.4 and Figure 6.5 for two different loss intervals. The investigated time interval is the same that is used for the event clustering.

It is immediately visible that for both stations, the wind clusters (station 13: 0, 1, 2, 3, 4; station 24: 1, 2, 3, 4) are strongly separated from the non-wind clusters. However, this is not surprising since the wind clusters are the only impulsive event classes found in both stations and the non-wind clusters consist solely of CW and thermal noise. Comparing the wind clusters of stations 13 and 24 to the combined test data of station 23, one can see that, except for cluster 3 of station 24, all wind clusters of both stations have much higher loss than the test data of station 23. For further comparison, the table of loss thresholds L and corresponding fractions of station 23 test wind data and cosmic ray simulations $\leq L$ (see Table 6.1) is extended by the fraction of wind events of station 13 and 24 equal or below different loss thresholds (see Table 6.2). Since the cosmic ray simulations are made for the full planned RNO-G station array instead of a particular station, it is possible to compare the cosmic ray simulations to stations 13 and 24 as well. Compared to



(a) **(b)**

Figure 6.4.: CDF of the minimum mean squared error (MSE) loss per cluster found for station 13 as well as the combined loss of test data of all station 23 wind clusters. The loss is calculated by taking the minimum of the MSE across all three channels. Cluster -1 contains all unclustered events and clusters 0, 1, 2, 3, and 4 are identified as wind clusters.



(a) **(b)**

Figure 6.5.: CDF of the minimum mean squared error (MSE) loss per cluster found for station 24 as well as the combined loss of test data of all station 23 wind clusters. The loss is calculated by taking the minimum of the MSE across all three channels. Cluster -1 contains all unclustered events and clusters 1, 2, 3, and 4 are identified as wind clusters.

L	station 23 % test data $\leq L$	station 13 % wind $\leq L$	station 24 % wind $\leq L$	CR simulations % 10σ CR $\leq L$
2.16	80.1	0.00	7.89	0.008
4.37	90.0	0.716	30.4	0.265
7.15	95.0	7.40	52.0	0.906
7.48	95.3	8.59	53.7	1.00
15.3	99.0	51.4	93.3	13.9

Table 6.2.: Thresholds L on the MSE loss and corresponding fractions (in percent) of station 23 test wind data, station 13 and 24 wind events, and cosmic ray (CR) simulations that are below or equal the threshold i.e. classified as wind. The simulations are produced for a 10σ trigger.

station 23, the fractions of wind events $\leq L$ are much lower for stations 13 and 24. This indicates that there are significant differences between some wind clusters of different stations, as suspected from the comparison of mean spectra in chapter 5. However, wind events of station 24 have an overall lower loss than wind events of station 13, so there might be more similarities between wind events of stations 23 and 24 than between stations 13 and 23.

Although no event clustering is done for stations 11, 12, 21, and 22, it is still possible to qualitatively compare those stations by examining the loss as a function of wind speed in a 2D histogram. The plots can be found in subsection A.4.3 in the appendix. Since force-triggered events are mostly thermal noise, only shallow-triggered events are shown in the plots. Note that all stations are evaluated on the discriminator that is trained exclusively on wind events of station 23. The plots show the raw distribution without any weighting or data-normalization. For stations 12, 13, 23, and 24, a blob-like structure at high loss and lower wind speeds can be observed, which contains mostly thermal noise events. Since thermal noise is independent of wind speed, the blob matches the distribution of wind speed in the investigated time period of two months, which is shown in Figure 5.7. The shape and position of this structure are, as one would expect, similar across the stations it occurs, since thermal noise should not be station-dependent. For the remaining stations, only a few events occur at this position. This is possibly due to the trigger being tuned in such a way that only a few thermal noise events satisfy the trigger condition of the shallow trigger. Additionally to the thermal noise blob, some stations show a second blob at lower loss and higher wind speeds. For stations 13 and 24, this is expected since wind events make up almost all impulsive background events in these two stations. Nevertheless, station 23 is the only station with a high fraction of events in the lowest loss region, indicating again

that there are differences between wind events of different stations. Whether these differences stem from different production processes, different arrival directions, or differences in signal processing of each station is not known.

All in all, these results suggest that, at least as long as the processes and origins of wind events are not known and understood, per-station data selection and training are necessary to reject wind events for all stations.

7. Summary, conclusions and outlook

In this thesis, wind events, a background class appearing during time periods of high wind speed, are investigated and identified for RNO-G. Firstly, clusters of wind events are identified and analyzed for several stations. After that, a neural network is trained to discriminate between wind events and non-wind events.

Since other backgrounds are present in the data and the physics of wind events is still unknown, a machine learning method (inspired by [39]) is used to first group similar events into clusters and then identify wind clusters. For this purpose, a variational autoencoder (VAE) model is trained on shallow-triggered events of the RNO-G station, for which clustering is desired. The VAE learns to represent events in a regularized low-dimensional latent feature space (bottleneck). Events are then clustered in this low-dimensional space using the clustering algorithm HDBSCAN. This procedure is done for stations 13, 23, and 24 for a time period of two months. However, the whole clustering process is not yet fully automated and requires a human to determine suitable HDBSCAN parameters. The data used in this thesis is only a small fraction (of the order of 5 %) of the total data recorded at Greenland due to satellite transfer speed limitations. Found clusters are then analyzed, primarily with respect to wind speed, to identify wind clusters.

Multiple wind clusters are found for the three investigated stations, whereby the number of events per cluster (cluster size) as well as the mean frequency spectra per cluster and the behavior of the cluster size as a function of the wind speed vary strongly across wind clusters within a station and across different stations. Additionally, the number of found wind clusters varies across different stations as well. Moreover, some wind clusters appear almost exclusively during a single high-wind day, while others appear during multiple high-wind periods. These differences in wind clusters within and across stations indicate that there could be different processes responsible for the production of wind event signals. Apart from wind speed, the behavior of clusters with respect to other weather quantities such as temperature, relative humidity, or wind direction is investigated as well. However, no conclusions can be drawn from the data about whether or not these quantities influence the production of certain wind event classes. The reason is that the investigated time period is rather short and it is not possible to disentangle wind

speed from other investigated quantities.

Furthermore, it is observable that for some wind clusters, the signal in a specific channel is often weaker than in the remaining two channels, which indicates that there is a preferred arrival direction for some clusters. This could mean that wind events of these clusters are produced at man-made structures, such as a station's solar panels. However, a detailed direction reconstruction of the events of found wind clusters is necessary to support this hypothesis.

Apart from wind event clusters and other known backgrounds, a yet unknown background class is found for station 23. This background class appears daily and only during daytime, but starts to disappear in September of 2022. This event class may stem from the station's solar panels, since a similar behavior, which is attributed to battery charging, is observed for other stations as well. Moreover, human activity during daytime could also be responsible for this background.

Since some wind clusters appear almost exclusively during a single high-wind day in the investigated time period of 2 months, it is suspected that the found wind clusters do not represent the full spectrum of wind events appearing over several years. Hence, more observation time is required to further study wind events. However, using more data may require training of a new VAE and another tuning of the HDBSCAN parameters, especially if new event classes arise. As mentioned before, the clustering process is not yet completely automated and therefore tuning the parameters by hand may become very time-consuming, in particular for more stations and significantly more data, which increases computation time. Thus, finding a way to make the clustering process more or even fully automatable is an important goal for future work.

Overall, the insights gained in this thesis do not suffice to understand how, where and under which conditions wind events are produced. Therefore, controlled experiments using e.g. a wind tunnel are required to understand the physics behind wind events in the future.

In the last part of this thesis, a method to discriminate between wind events and non-wind events is developed to reduce a potential interference of wind events with other analyses. Here, the anomaly detection technique is chosen, where another VAE is trained exclusively on wind events of station 23. Events, that are of the same class as the ones the VAE is trained on, should then show a lower reconstruction loss than events of classes, which are unknown to the VAE. By choosing a threshold on the loss, one obtains a binary classifier, where events with a loss below or equal to the threshold are classified as wind. This threshold can be chosen for each individual analysis. For later evaluation of wind clusters of station 23, only the test data sets are used. However, only ≈ 5000 and ≈ 1700 unique events are available for training and validation.

A comparison of the CDFs of the loss of individual clusters of station 23 shows

that the majority of non-wind events indeed have a higher loss than the majority of wind events. For some wind clusters, however, there is a considerable overlap with some non-wind clusters. Therefore, if a very pure set of wind events is desired, a low loss threshold must be chosen. This overlap is expected to decrease if more training data is available.

An example of a physics analysis that uses the shallow part of an RNO-G station is the search for cosmic ray events. However, wind events can interfere with this analysis, making the search for cosmic rays during a high-wind period very difficult. Therefore, the loss of the combined test data of wind events of station 23 is compared to the loss of cosmic ray simulations to investigate if a wind discriminator can provide a benefit for a cosmic ray search. Indeed, a strong separation of test data from cosmic ray simulations is observed. It is for example possible to correctly classify $\approx 95\,\%$ of the test data while falsely classifying only $\approx 1\,\%$ of the cosmic ray simulations. This indicates that a wind discriminator can indeed be very helpful for a cosmic ray search. However, the test data might not necessarily represent all wind events found in the investigated time period, since the unclustered events (cluster -1) contain a considerable fraction of wind-correlated events as well. Since cluster -1 is not used for training, validation, and testing, some of the unclustered wind events may have a high loss. This might reduce the fraction of all wind events equal to or below the threshold and thus decrease the effectiveness of the discriminator. Therefore, it is important to find a way to reduce the number of wind-correlated events in cluster -1 in the future. A first step in this direction could be to further optimize the parameters of the clustering algorithm HDBSCAN. Additionally, one could try to find clusters inside the unclustered events by running the clustering algorithm separately on the unclustered events.

Finally, the wind discriminator, which is trained exclusively on wind events of station 23, is evaluated on other stations as well. Since event clustering is performed on stations 13 and 24, wind clusters of stations 13, 23, and 24 can be compared directly. The comparison of clusters of stations 13 and 24 with the combined test data of station 23 shows that the wind clusters of stations 13 and 24 have mostly a higher loss than the wind clusters of station 23. This means that the differences in wind events of different stations are so large that the VAE is not able to accurately reconstruct wind events of other stations. A comparison of the loss as a function of the wind speed for all deployed stations comes to the same conclusion, as only station 23 shows a considerable fraction of events at very low loss and high wind speeds. Why wind events of different stations are so different is not known. Hence, as long as the physics of wind events are not understood, a separate wind discriminator is required for each station. As a result, individual event clustering and identification of wind clusters have to be performed per station to obtain training data. Since the clustering process is not yet fully

automated, this task can become very time-consuming, considering that in total 35 stations are planned for RNO-G. Therefore, as pointed out before, finding a way to fully automate the clustering process would be an important subject of future work. Additionally, it would be useful to find a way to prevent the production of wind events directly in the field. If, for example, wind events originate from metal structures, e.g. parts of the solar panels of a station, one could wrap such structures in e.g. a non-metal foil to prevent contact between metal and wind.

Although it is overall possible to separate high fractions of wind events from non-wind events, the results are not perfect and multiple aspects of the method could be improved. For example, a different set of hyperparameters could provide more separation and less overlap between wind and non-wind events. Additionally, different discrimination metrics instead of the mean squared error might lead to further improvements. It might also be useful to investigate different metrics for different analyses. Apart from metrics that incorporate the reconstruction error, one could consider metrics that also take the latent space distribution into account. As pointed out before, only a small fraction of the recorded data is used and thus the use of more data for training is expected to improve the performance of the discriminator significantly. Moreover, a different VAE architecture and/or a different data preparation method might increase discrimination performance. One could, for example, try to use the time series (waveform) instead of the spectrogram of an event and use a VAE built from LSTM [47] layers, which have a memory and are therefore suitable for time series inputs. In this case, it might also be useful to investigate denoising techniques to reduce thermal noise in the waveforms. Furthermore, the sampling rate of the waveforms is planned to be reduced in the future, so that the waveforms become longer. This, in combination with further adjustments to e.g. the cable delays, could allow to see the shallow-triggered signals in the deeper antennas as well, which is currently not possible. Then, one could use more antennas for the wind discrimination training, which might allow a better distinction between wind and non-wind events. In addition to changes made to the waveforms, the number of upward-facing antennas might be changed to four in the future. This would increase the number of upward-facing LPDAs by one and thus more waveforms i.e. more data could be available per event.

Bibliography

[1] KATRIN collaboration. "Direct neutrino-mass measurement with sub-electronvolt sensitivity". In: *Nature Physics* 18.2 (Feb. 2022), pp. 160–166. ISSN: 1745-2481. DOI: 10.1038/s41567-021-01463-1.

[2] IceCube collaboration. "The IceCube Neutrino Observatory: instrumentation and online systems". In: *Journal of Instrumentation* 12.03 (Mar. 2017), P03012. DOI: 10.1088/1748-0221/12/03/P03012.

[3] KM3NeT collaboration. "Letter of intent for KM3NeT 2.0". In: *J. Phys. G* 43.8 (2016), p. 084001. DOI: 10.1088/0954-3899/43/8/084001.

[4] RNO-G collaboration. "Design and sensitivity of the Radio Neutrino Observatory in Greenland (RNO-G)". In: *Journal of Instrumentation* 16.03 (Mar. 2021), P03025. DOI: 10.1088/1748-0221/16/03/p03025.

[5] RNO-G collaboration. "In situ, broadband measurement of the radio frequency attenuation length at Summit Station, Greenland". In: *Journal of Glaciology* 68.272 (2022), pp. 1234–1242. DOI: 10.1017/jog.2022.40.

[6] S. Hallmann for the RNO-G collaboration. "Status and recent results from the Radio Neutrino Observatory in Greenland (RNO-G)". In: *Proceedings of 38th International Cosmic Ray Conference — PoS(ICRC2023)*. Vol. 444. 2023, p. 1043. DOI: 10.22323/1.444.1043.

[7] J.A. Aguilar et al. "Triboelectric backgrounds to radio-based polar ultra-high energy neutrino (UHEN) experiments". In: *Astroparticle Physics* 145 (Mar. 2023), p. 102790. DOI: 10.1016/j.astropartphys.2022.102790.

[8] G. A. Askar'yan. "Excess negative charge of an electron-photon shower and its coherent radio emission". In: *Zh. Eksp. Teor. Fiz.* 41 (1961), pp. 616–618.

[9] L. Pyras and I. Plaisier for the RNO-G collaboration. "The Radio Neutrino Observatory Greenland: Status Update and Prospect for Air Showers". In: *Proceedings of 27th European Cosmic Ray Symposium — PoS(ECRS)*. Vol. 423. 2023, p. 088. DOI: 10.22323/1.423.0088.

[10] K. J. Astrom and T. Hagglund. *PID controllers: Theory, Design and Tuning*. 2nd ed. ISA, June 1995.

[11] C. Glaser et al. "NuRadioReco: a reconstruction framework for radio neutrino detectors". In: *The European Physical Journal C* 79.6 (June 2019). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-019-6971-5.

[12] M. Reichert. *The diffuse Galactic signal in RNO-G*. Bachelor's thesis. 2023. URL: https://ecap.nat.fau.de/wp-content/uploads/2023/09/Bachelorarbeit_Moritz_Reichert.pdf.

[13] ARIANNA collaboration. "Radio detection of air showers with the ARIANNA experiment on the Ross Ice Shelf". In: *Astroparticle Physics* 90 (2017), pp. 50–68. ISSN: 0927-6505. DOI: 10.1016/j.astropartphys.2017.02.003.

[14] T. Markus et al. "The Ice, Cloud, and land Elevation Satellite-2 (ICESat-2): Science requirements, concept, and implementation". In: *Remote Sensing of Environment* 190 (2017), pp. 260–273. ISSN: 0034-4257. DOI: 10.1016/j.rse.2016.12.029.

[15] L. Pyras et al. "Ultra-high energy muons in radio neutrino detectors". In: *Proceedings of 38th International Cosmic Ray Conference — PoS(ICRC2023)*. Vol. 444. 2023, p. 1076. DOI: 10.22323/1.444.1076.

[16] G. C. Simpson. *British Antarctic Expedition 1910-1913: meteorology*. Thacker, Spink & Company, 1919.

[17] D. S. Schmidt, R. A. Schmidt, and J. D. Dent. "Electrostatic Force in Blowing Snow". In: *Boundary-Layer Meteorology* 93.1 (Oct. 1999), pp. 29–45. DOI: 10.1023/a:1002045818907.

[18] M. Gordon and P. A. Taylor. "The Electric Field During Blowing Snow Events". In: *Boundary-Layer Meteorology* 130.1 (Nov. 2008), pp. 97–115. DOI: 10.1007/s10546-008-9333-7.

[19] J. Latham and B. J. Mason. "Electric charge transfer associated with temperature gradients in ice". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 260.1303 (Mar. 1961), pp. 523–536. DOI: 10.1098/rspa.1961.0051.

[20] J. E. Nanevicz. "Static Charging and Its Effects on Avionic Systems". In: *IEEE Transactions on Electromagnetic Compatibility* EMC-24.2 (1982), pp. 203–209. DOI: 10.1109/TEMC.1982.304031.

[21] F. Marquardt. "Machine learning and quantum devices". In: *SciPost Physics Lecture Notes* (May 2021). DOI: 10.21468/scipostphyslectnotes.29.

[22] R. Abdulkadirov, P. Lyakhov, and N. Nagornov. "Survey of Optimization Algorithms in Modern Neural Networks". In: *Mathematics* 11.11 (2023). ISSN: 2227-7390. DOI: 10.3390/math11112466.

[23] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR*. 2015. arXiv: 1412.6980 [cs.LG].

[24] V. Dumoulin and F. Visin. *A guide to convolution arithmetic for deep learning*. 2018. arXiv: 1603.07285v2 [stat.ML].

[25] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (visited on 09/26/2023).

[26] F. Chollet et al. *Keras*. 2015. URL: https://keras.io (visited on 09/26/2023).

[27] T. O'Malley et al. *KerasTuner*. 2019. URL: https://github.com/keras-team/keras-tuner (visited on 09/26/2023).

[28] R. J. G. B. Campello, D. Moulavi, and J. Sander. "Density-Based Clustering Based on Hierarchical Density Estimates". In: *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2013, pp. 160–172. DOI: 10.1007/978-3-642-37456-2_14.

[29] L. McInnes, J. Healy, and S. Astels. *How HDBSCAN Works*. URL: https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html (visited on 11/22/2023).

[30] L. McInnes, J. Healy, and S. Astels. "hdbscan: Hierarchical density based clustering". In: *The Journal of Open Source Software* 2.11 (Mar. 2017). DOI: 10.21105/joss.00205.

[31] L. McInnes, J. Healy, and J. Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML].

[32] L. McInnes et al. "UMAP: Uniform Manifold Approximation and Projection". In: *The Journal of Open Source Software* 3.29 (2018), p. 861. DOI: 10.21105/joss.00861.

[33] G. Pang et al. "Deep Learning for Anomaly Detection: A Review". In: *ACM Comput. Surv.* 54.2 (Mar. 2021). ISSN: 0360-0300. DOI: 10.1145/3439950.

[34] D. P Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR*. 2014. arXiv: 1312.6114 [stat.ML].

[35] D. P. Kingma and M. Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392. DOI: 10.1561/2200000056.

[36] S. Odaibo. *Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function*. 2019. arXiv: 1907.08956v1 [cs.LG].

[37] I. Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *5th International Conference on Learning Representations, ICLR*. 2017. URL: https://openreview.net/forum?id=Sy2fzU9gl.

[38] National Oceanic and Atmospheric Administration, Oceanic and Atmospheric Research, Global Monitoring Laboratory. URL: https://gml.noaa.gov/aftp/data/meteorology/in-situ/sum/ (visited on 09/25/2023).

[39] T. Glüsenkamp for the RNO-G collaboration. "VAE-based latent-space classification of RNO-G data". In: *Proceedings of 38th International Cosmic Ray Conference — PoS(ICRC2023)*. Vol. 444. 2023, p. 1056. DOI: 10.22323/1.444.1056.

[40] P. Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[41] F. Chollet. *Variational AutoEncoder*. 2020. URL: https://keras.io/examples/generative/vae/ (visited on 09/26/2023).

[42] L. Pyras. *Optimizing the triggering strategy for the detection of cosmic rays with the Radio Neutrino Observatory Greenland (RNO-G)*. Master's thesis. 2020. DOI: 10.3204/PUBDB-2021-01462.

[43] J. Henrichs and A. Nelles for the RNO-G collaboration. "Searching for cosmic-ray air showers with RNO-G". In: *Proceedings of 38th International Cosmic Ray Conference — PoS(ICRC2023)*. Vol. 444. 2023, p. 259. DOI: 10.22323/1.444.0259.

[44] D. Heck et al. *CORSIKA: A Monte Carlo code to simulate extensive air showers*. Tech. rep. FZKA-6019. 1998. DOI: 10.5445/IR/270043064.

[45] T. Huege, M. Ludwig, and C. W. James. "Simulating radio emission from air showers with CoREAS". In: *AIP Conference Proceedings* 1535.1 (May 2013), pp. 128–132. DOI: 10.1063/1.4807534.

[46] V. Verzi for the Pierre Auger collaboration. "Measurement of the energy spectrum of ultra-high energy cosmic rays using the Pierre Auger Observatory". In: *Proceedings of 36th International Cosmic Ray Conference — PoS(ICRC2019)*. Vol. 358. 2019, p. 450. DOI: 10.22323/1.358.0450.

[47] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

A. Appendix

A.1. Trigger rates of stations 12, 23, and 24

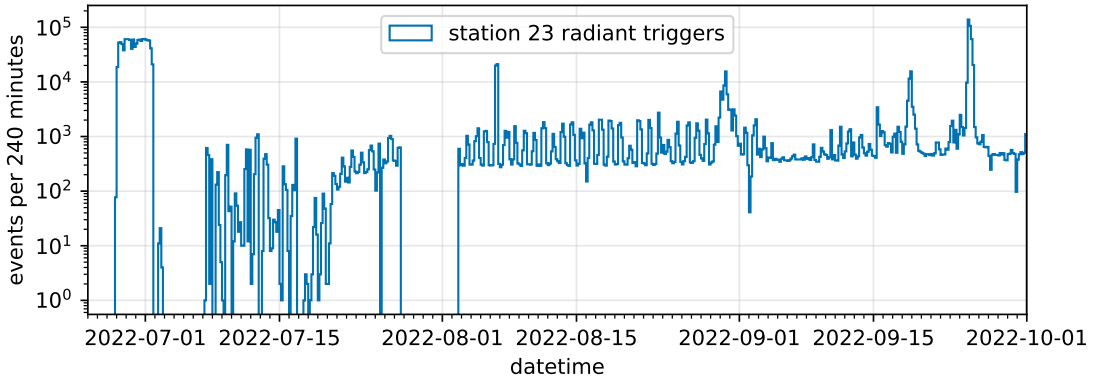


Figure A.1.: Station 23: shallow (radiant) trigger rates between 2022-06-25 and 2022-10-01. Time is given in UTC.

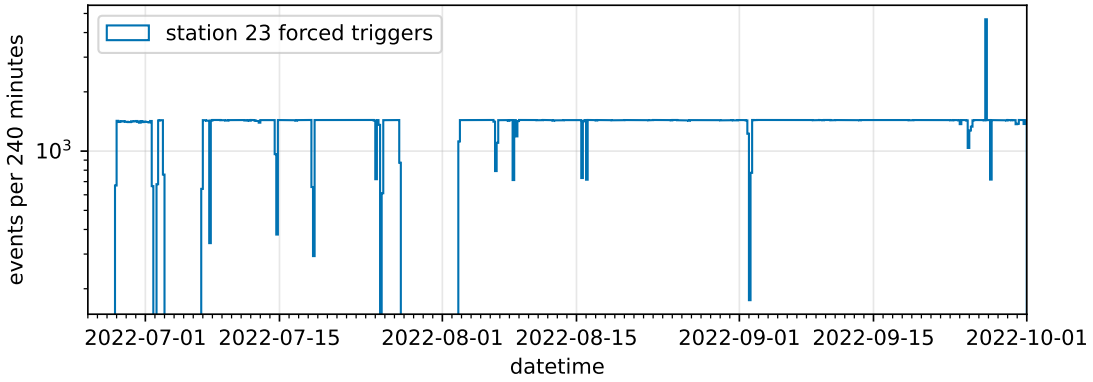


Figure A.2.: Station 23: force trigger rates between 2022-06-25 and 2022-10-01. Time is given in UTC.

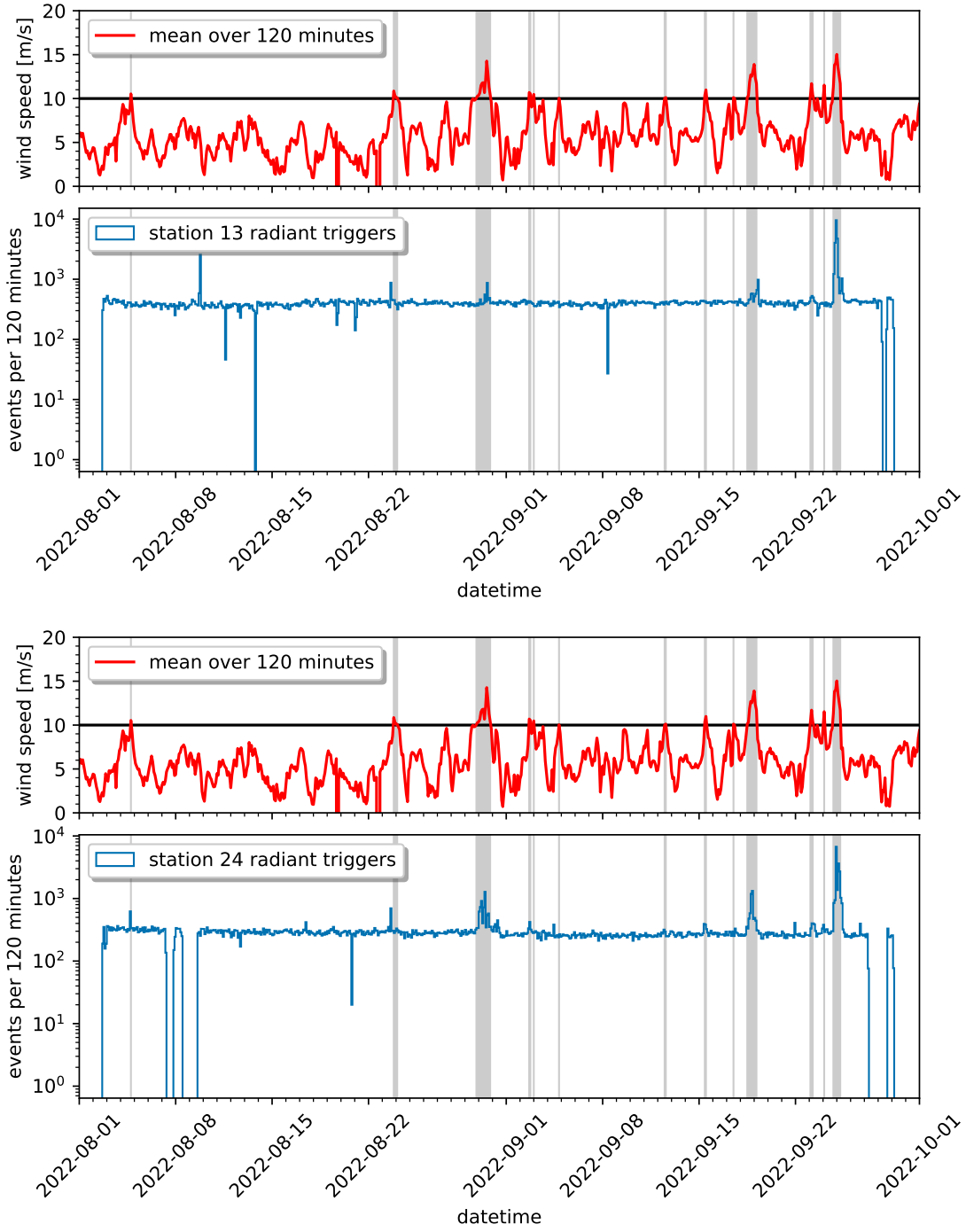


Figure A.3.: Shallow (radiant) trigger rates (in counts per 120 minutes, logarithmic scale) and the mean wind speed over 120 minutes for stations 13 (top) and 24 (bottom). Time intervals where the mean wind speed is greater or equal to 10 m/s (marked by a horizontal line) are shaded in gray. Time is given in UTC.

A.2. Event clustering using a variational autoencoder

A.2.1. Model architecture

Layer	Name	Output shape	Input
InputLayer	`input_1`	(128, 128, 3)	-
GaussianNoise	`gaussian_noise`	(128, 128, 3)	`input_1`
Conv2D + LReLU	`conv_1_1`	(128, 128, 16)	`gaussian_noise`
Conv2D + LReLU	`conv_1_2`	(64, 64, 16)	`conv_1_1`
Dropout	`dropout_1`	(64, 64, 16)	`conv_1_2`
Conv2D + LReLU	`conv_2_1`	(64, 64, 32)	`dropout_1`
Conv2D + LReLU	`conv_2_2`	(32, 32, 32)	`conv_2_1`
Dropout	`dropout_2`	(32, 32, 32)	`conv_2_2`
Conv2D + LReLU	`conv_3_1`	(32, 32, 64)	`dropout_2`
Conv2D + LReLU	`conv_3_2`	(16, 16, 64)	`conv_3_1`
Dropout	`dropout_3`	(16, 16, 64)	`conv_3_2`
Conv2D + LReLU	`conv_4_1`	(16, 16, 128)	`dropout_3`
Conv2D + LReLU	`conv_4_2`	(8, 8, 128)	`conv_4_1`
Dropout	`dropout_4`	(8, 8, 128)	`conv_4_2`
Conv2D + LReLU	`conv_5_1`	(8, 8, 256)	`dropout_4`
Conv2D + LReLU	`conv_5_2`	(4, 4, 256)	`conv_5_1`
Flatten	`flatten`	(4096)	`conv_5_2`
Dense + LReLU	`dense_1`	(64)	`flatten`
Dense + Linear	`z_mean`	(9)	`dense_1`
Dense + Linear	`z_log_var`	(9)	`dense_1`
Sampling	`z`	(9)	`z_mean` + `z_log_var`

Table A.1.: Structure of the encoder of the VAE used for event clustering. The first column specifies the respective layer and its corresponding activation function if the layer has one. LReLU denotes the LeakyReLU activation. The sampling layer calculates the latent variable z using the outputs of the `z_mean` and `z_log_var` layers and a random number generator according to the reparameterization trick (see Equation 4.27). For a 2D convolutional layer, the first two dimensions of the output shape denote the spatial dimensions and the last dimension denotes the number of channels/filters.

Layer	Name	Output shape	Input
InputLayer	input_2	(9)	z
Dense + LReLU	dense_2	(4096)	input_2
Reshape	reshape	(4, 4, 256)	dense_2
Dropout	dropout_5	(4, 4, 256)	reshape
Conv2DTr + LReLU	convtr_1_1	(4, 4, 256)	dropout_5
Conv2DTr + LReLU	convtr_1_2	(8, 8, 256)	convtr_1_1
Dropout	dropout_6	(8, 8, 256)	convtr_1_2
Conv2DTr + LReLU	convtr_2_1	(8, 8, 128)	dropout_6
Conv2DTr + LReLU	convtr_2_2	(16, 16, 128)	convtr_2_1
Dropout	dropout_7	(16, 16, 128)	convtr_2_2
Conv2DTr + LReLU	convtr_3_1	(16, 16, 64)	dropout_7
Conv2DTr + LReLU	convtr_3_2	(32, 32, 64)	convtr_3_1
Dropout	dropout_8	(32, 32, 64)	convtr_3_2
Conv2DTr + LReLU	convtr_4_1	(32, 32, 32)	dropout_8
Conv2DTr + LReLU	convtr_4_2	(64, 64, 32)	convtr_4_1
Dropout	dropout_9	(64, 64, 32)	convtr_4_2
Conv2DTr + LReLU	convtr_5_1	(64, 64, 16)	dropout_9
Conv2DTr + LReLU	convtr_5_2	(128, 128, 16)	convtr_5_1
Conv2DTr + max(ReLU, 1)	output	(128, 128, 3)	convtr_5_2

Table A.2.: Structure of the decoder of the VAE- used for event clustering. The first column specifies the respective layer and its corresponding activation function if the layer has one. LReLU denotes the LeakyReLU activation and Conv2DTr means 2D transposed convolutional layer. The activation function of the output layer is a ReLU limited to 1. For a 2D convolutional layer, the first two dimensions of the output shape denote the spatial dimensions and the last dimension denotes the number of channels/filters.

A.2.2. Clustering results: station 23

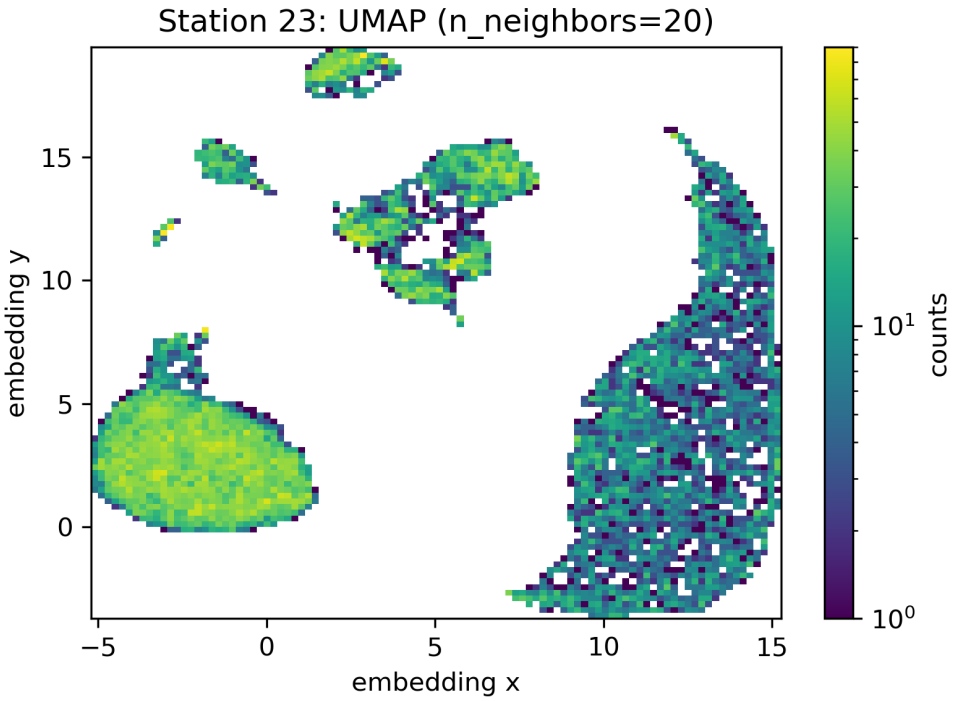


Figure A.4.: 2D histogram to visualize the density of the UMAP representation of latent vectors of station 23 shallow- and force-triggered events.

parameter	value
`min_samples`	1
`min_cluster_size`	50
`cluster_selection_method`	'leaf'
`cluster_selection_epsilon`	0.615
`leaf_size`	1

Table A.3.: Non-default HDBSCAN parameters used for the final clustering for station 23.

A.2.2.1. CDF of cluster size as a function of different weather quantities

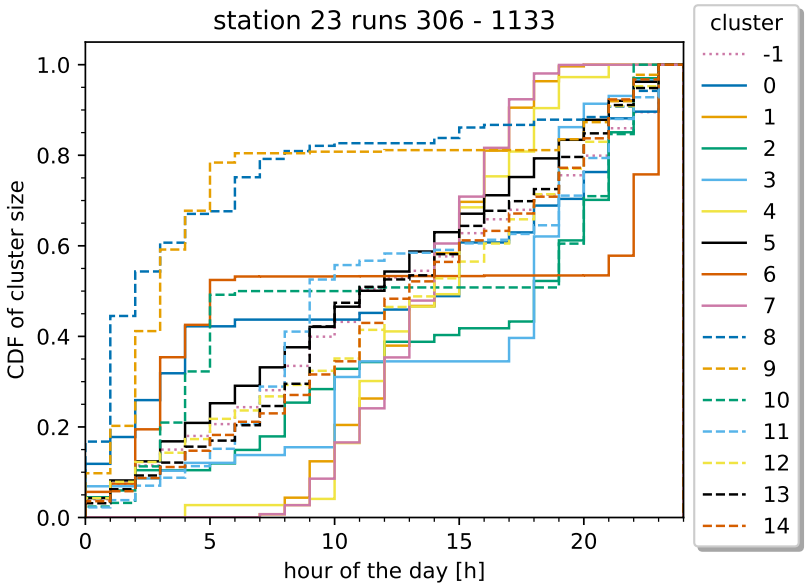


Figure A.5.: Cumulative distribution of the cluster size as a function of the hour of the day for station 23.

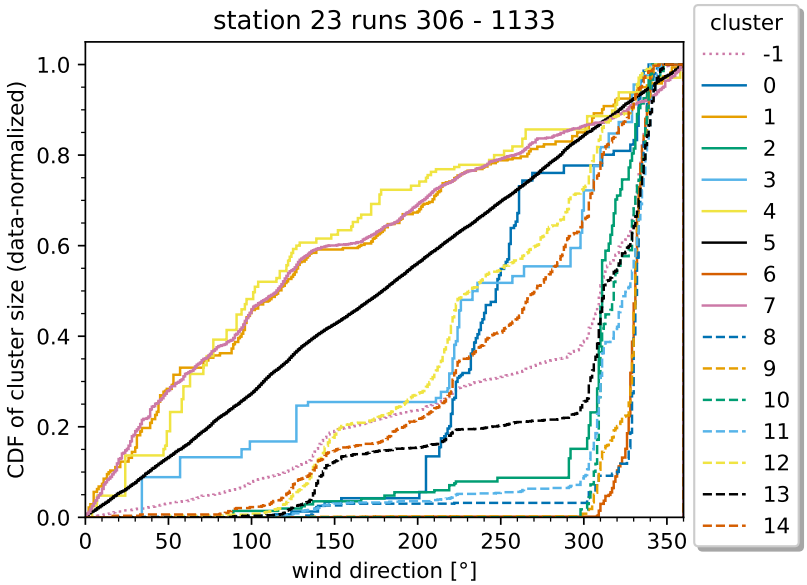


Figure A.6.: Cumulative distribution (data-normalized) of the cluster size as a function of the wind direction for station 23.

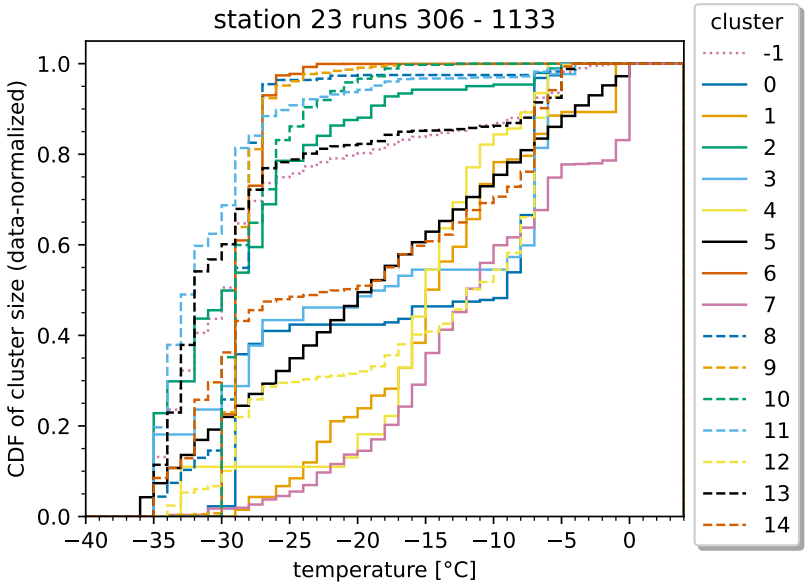


Figure A.7.: Cumulative distribution (data-normalized) of the cluster size as a function of the temperature for station 23.

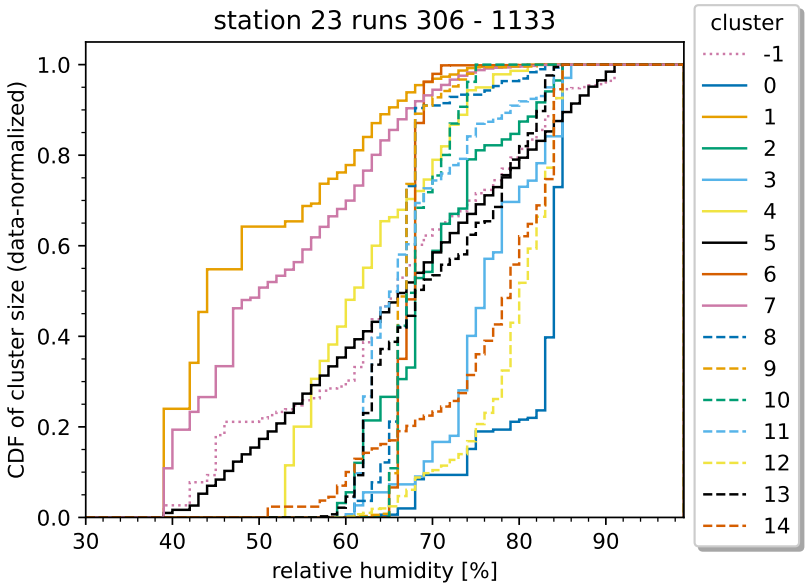


Figure A.8.: Cumulative distribution (data-normalized) of the cluster size as a function of the relative humidity for station 23.

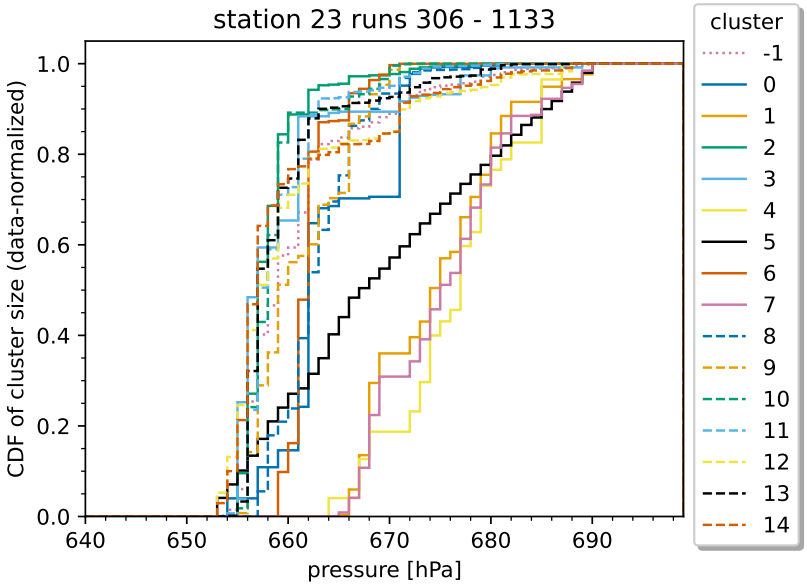
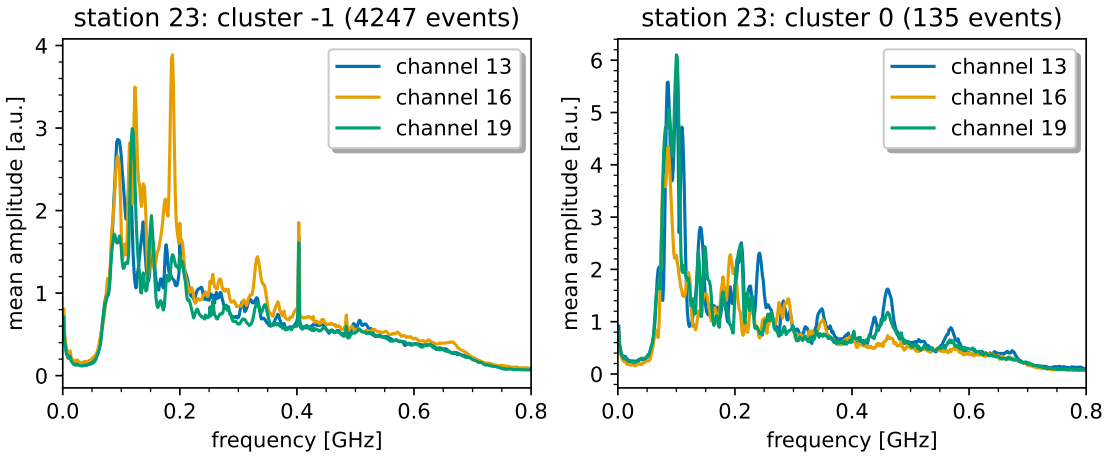


Figure A.9.: Cumulative distribution (data-normalized) of the cluster size as a function of the barometric pressure for station 23.

A.2.2.2. Mean frequency spectra per cluster



(a) **(b)**

Figure A.10.: Mean frequency spectra per channel for a) cluster -1 and b) cluster 0 (station 23).

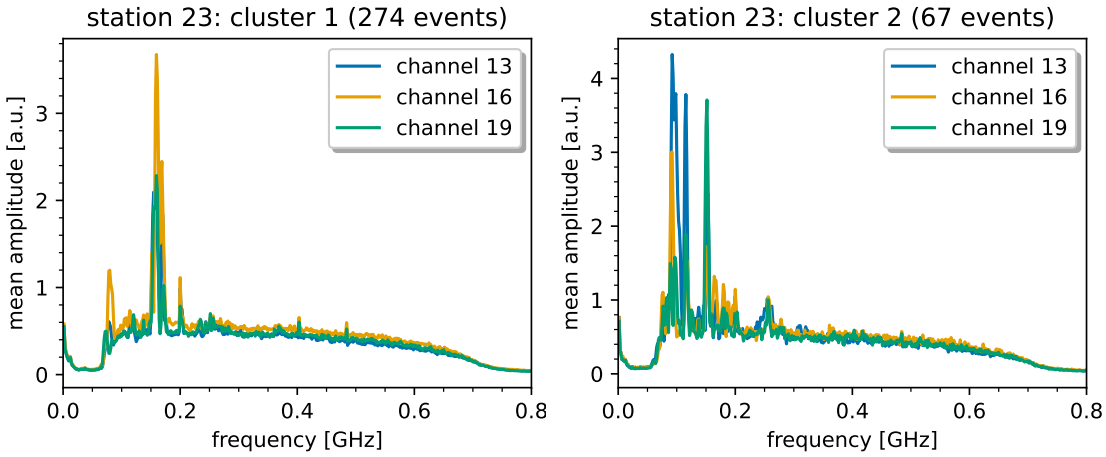


Figure A.11.: Mean frequency spectra per channel for a) cluster 1 and b) cluster 2 (station 23).

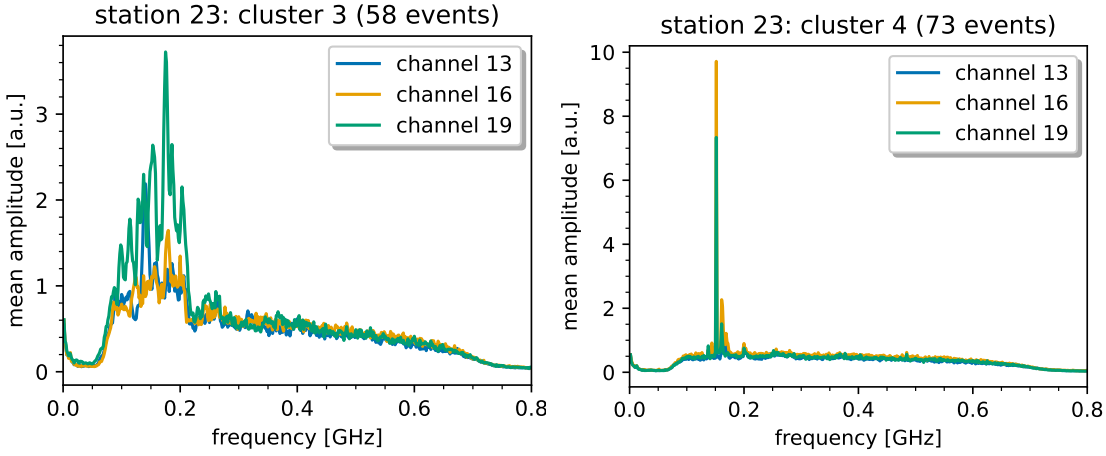
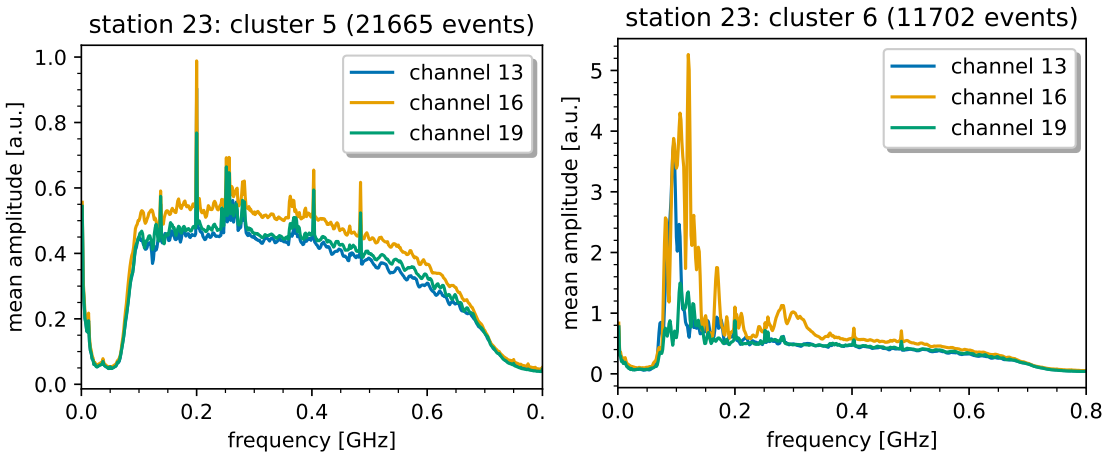
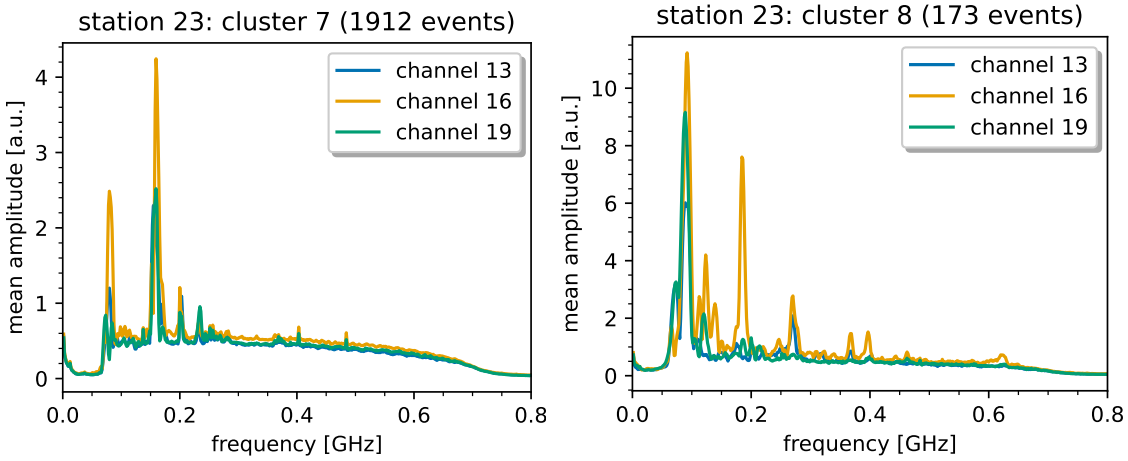


Figure A.12.: Mean frequency spectra per channel for a) cluster 3 and b) cluster 4 (station 23).



(a) **(b)**

Figure A.13.: Mean frequency spectra per channel for a) cluster 5 and b) cluster 6 (station 23).



(a) **(b)**

Figure A.14.: Mean frequency spectra per channel for a) cluster 7 and b) cluster 8 (station 23).

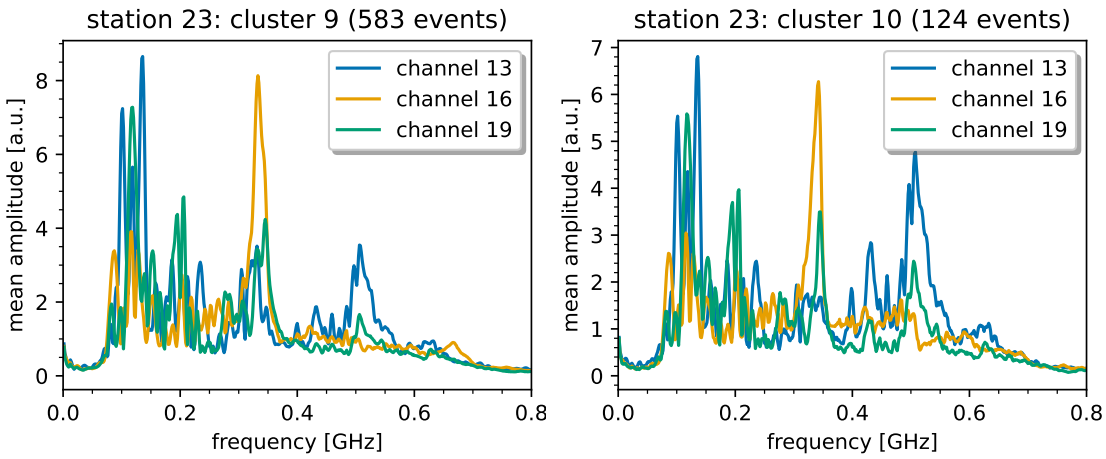


Figure A.15.: Mean frequency spectra per channel for a) cluster 9 and b) cluster 10 (station 23).

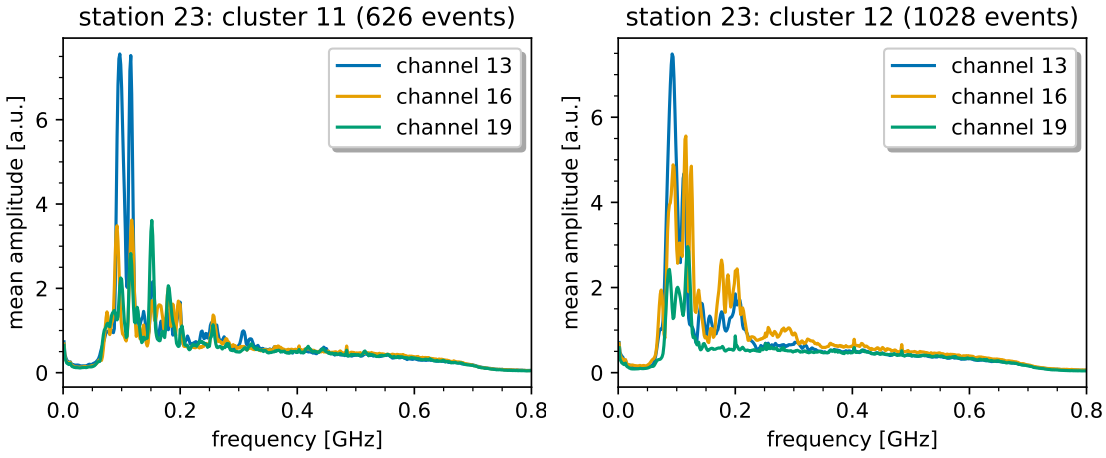
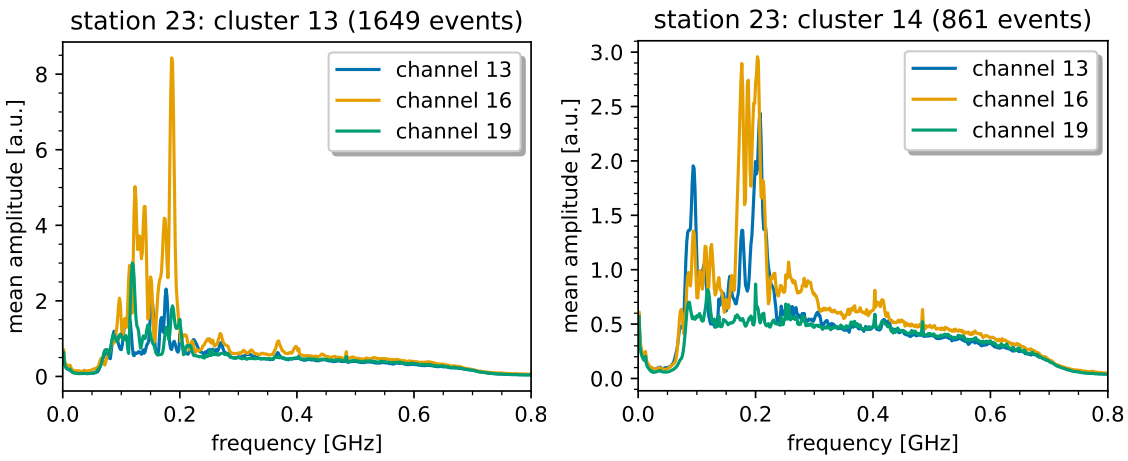


Figure A.16.: Mean frequency spectra per channel for a) cluster 11 and b) cluster 12 (station 23).



(a) **(b)**

Figure A.17.: Mean frequency spectra per channel for a) cluster 13 and b) cluster 14 (station 23).

A.2.2.3. Example waveforms and spectra per cluster

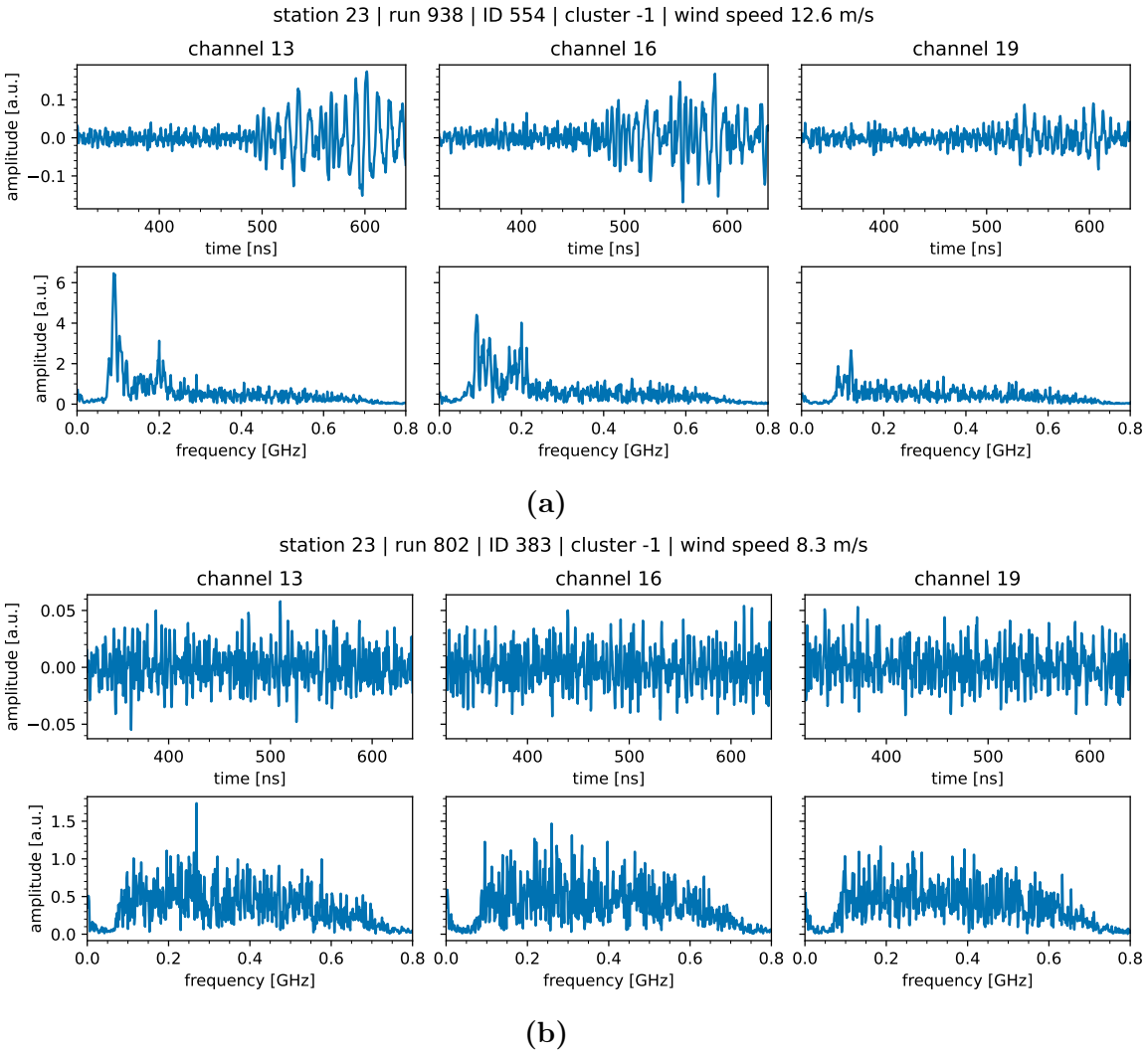
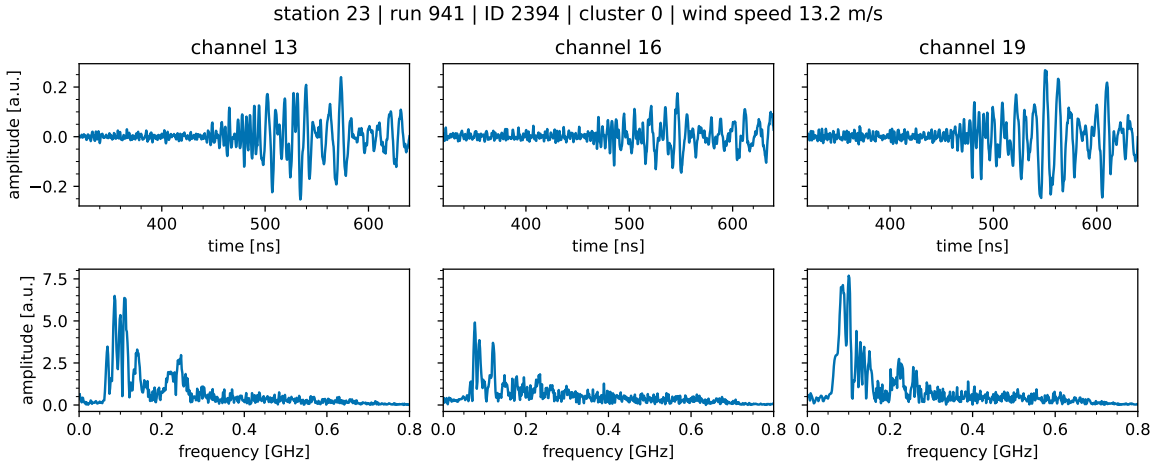
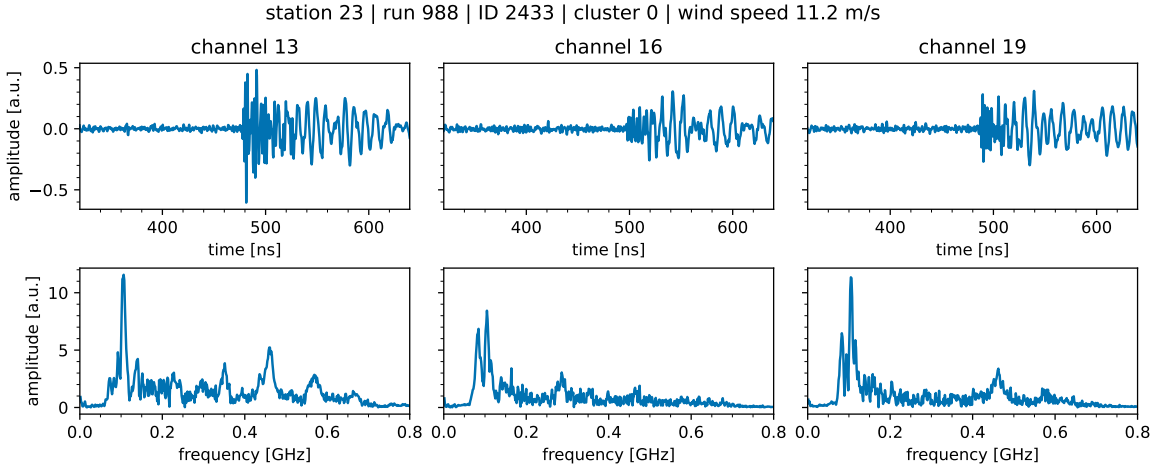


Figure A.18.: Example waveforms and frequency spectra for cluster -1 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.



(a)



(b)

Figure A.19.: Example waveforms and frequency spectra for cluster 0 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

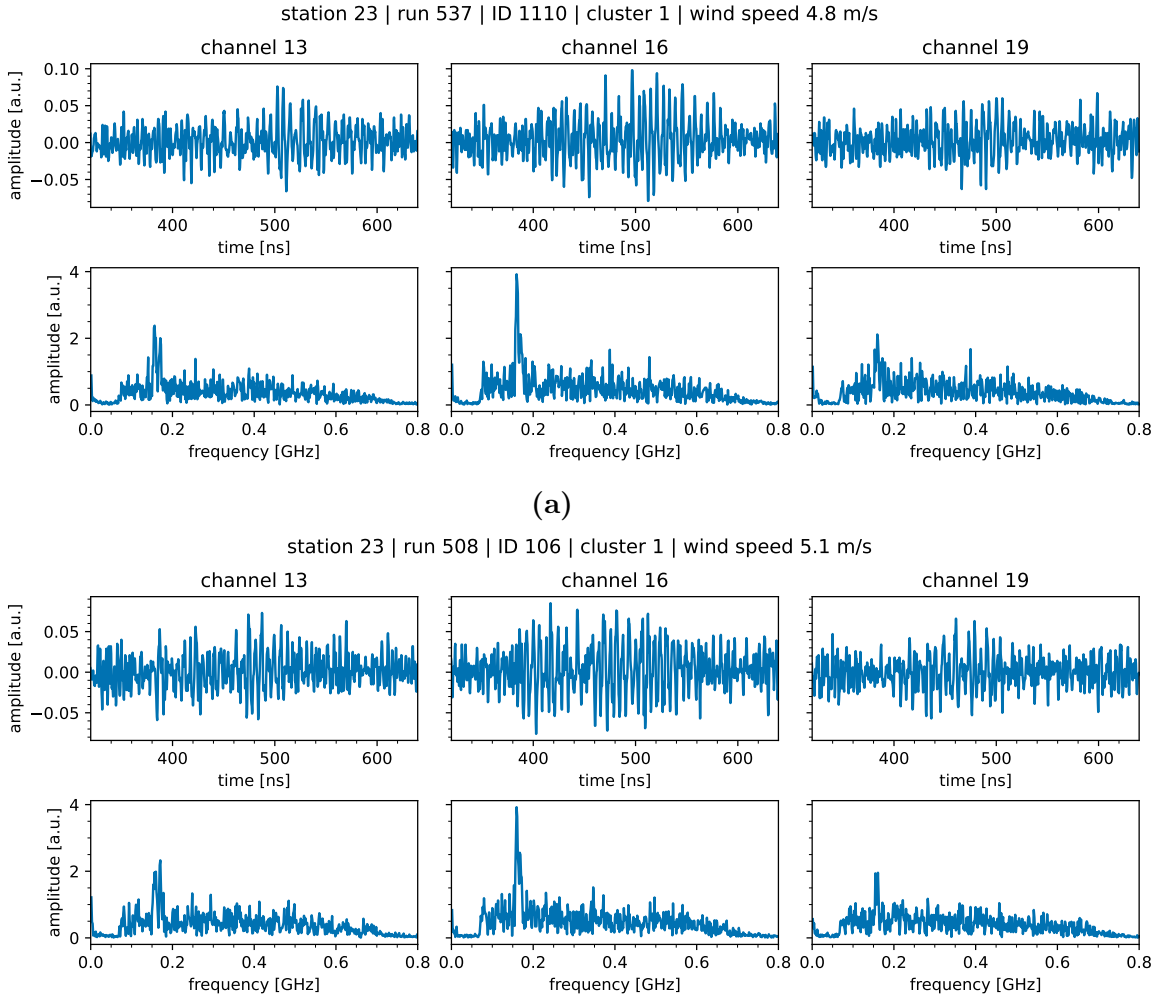


Figure A.20.: Example waveforms and frequency spectra for cluster 1 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

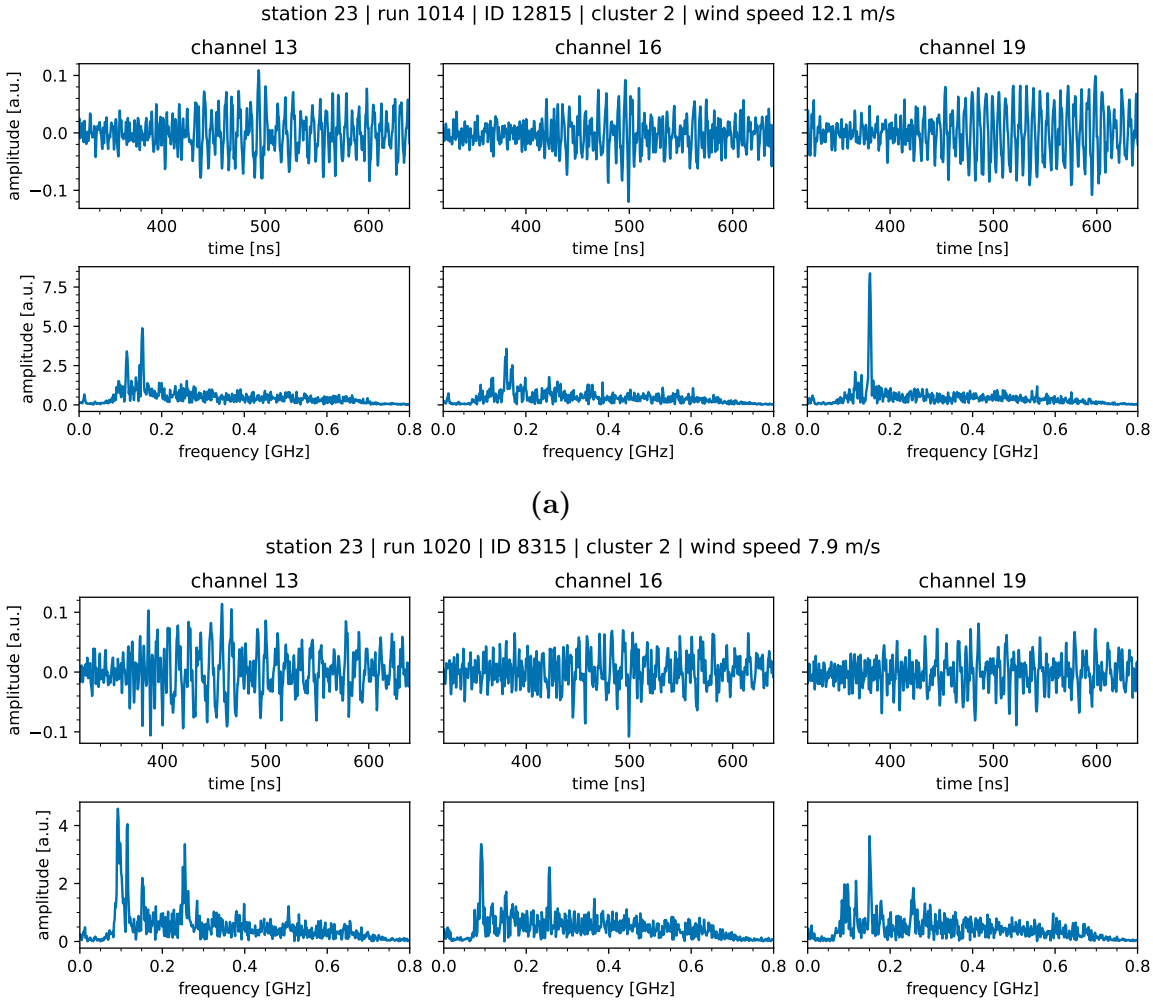
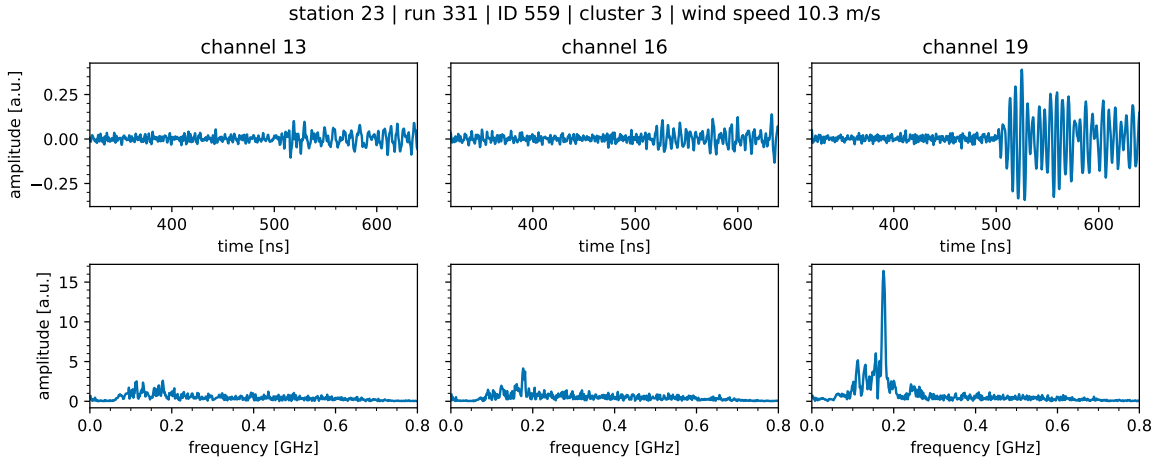
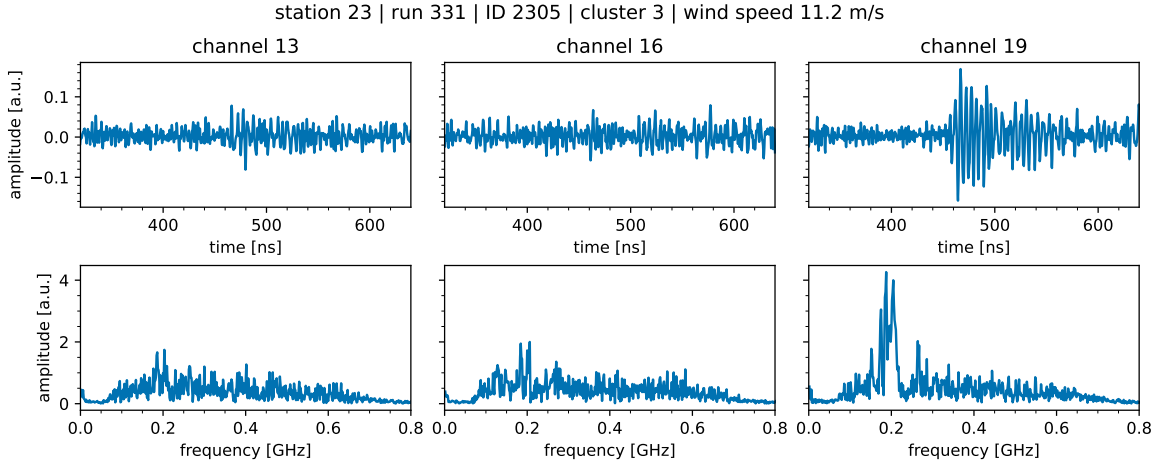


Figure A.21.: Example waveforms and frequency spectra for cluster 2 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

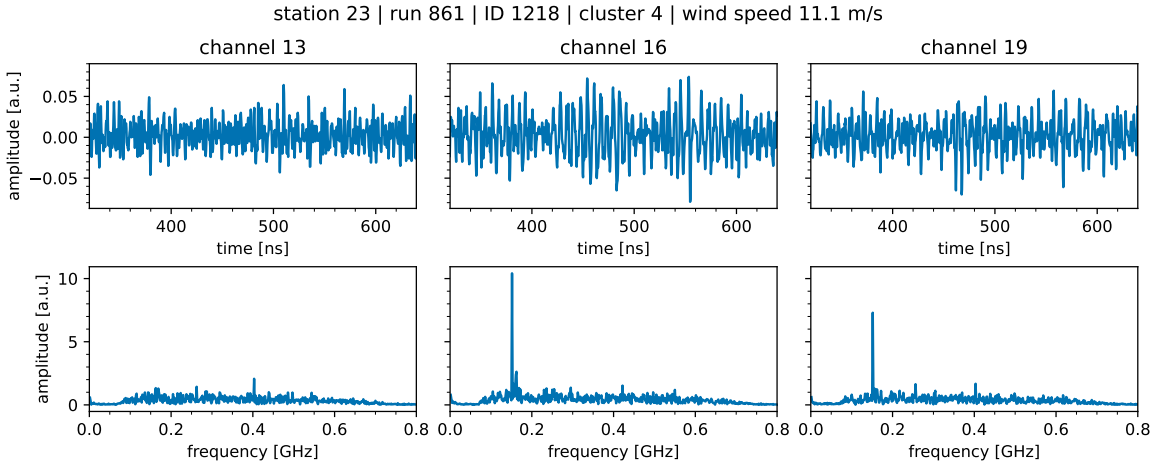


(a)

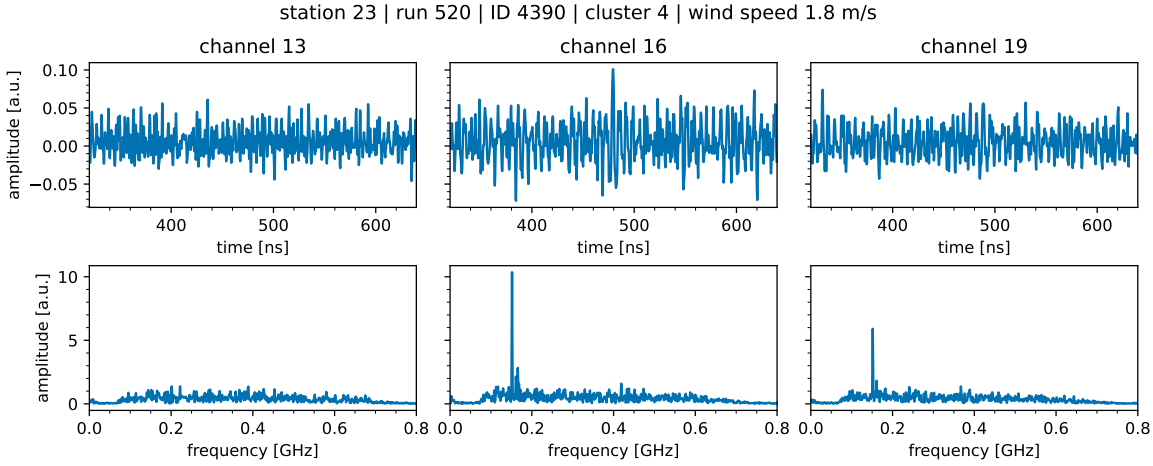


(b)

Figure A.22.: Example waveforms and frequency spectra for cluster 3 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.



(a)



(b)

Figure A.23.: Example waveforms and frequency spectra for cluster 4 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

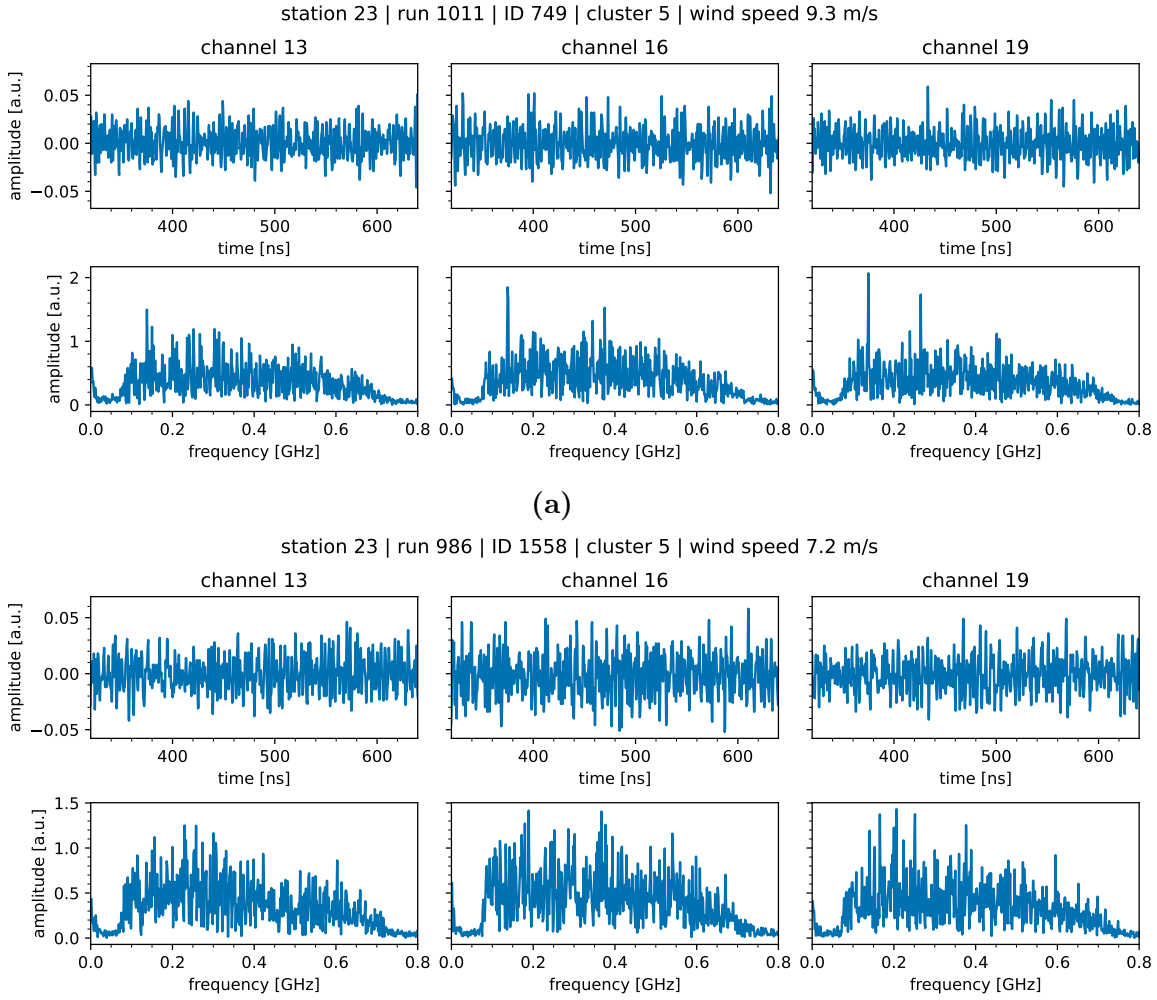


Figure A.24.: Example waveforms and frequency spectra for cluster 5 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

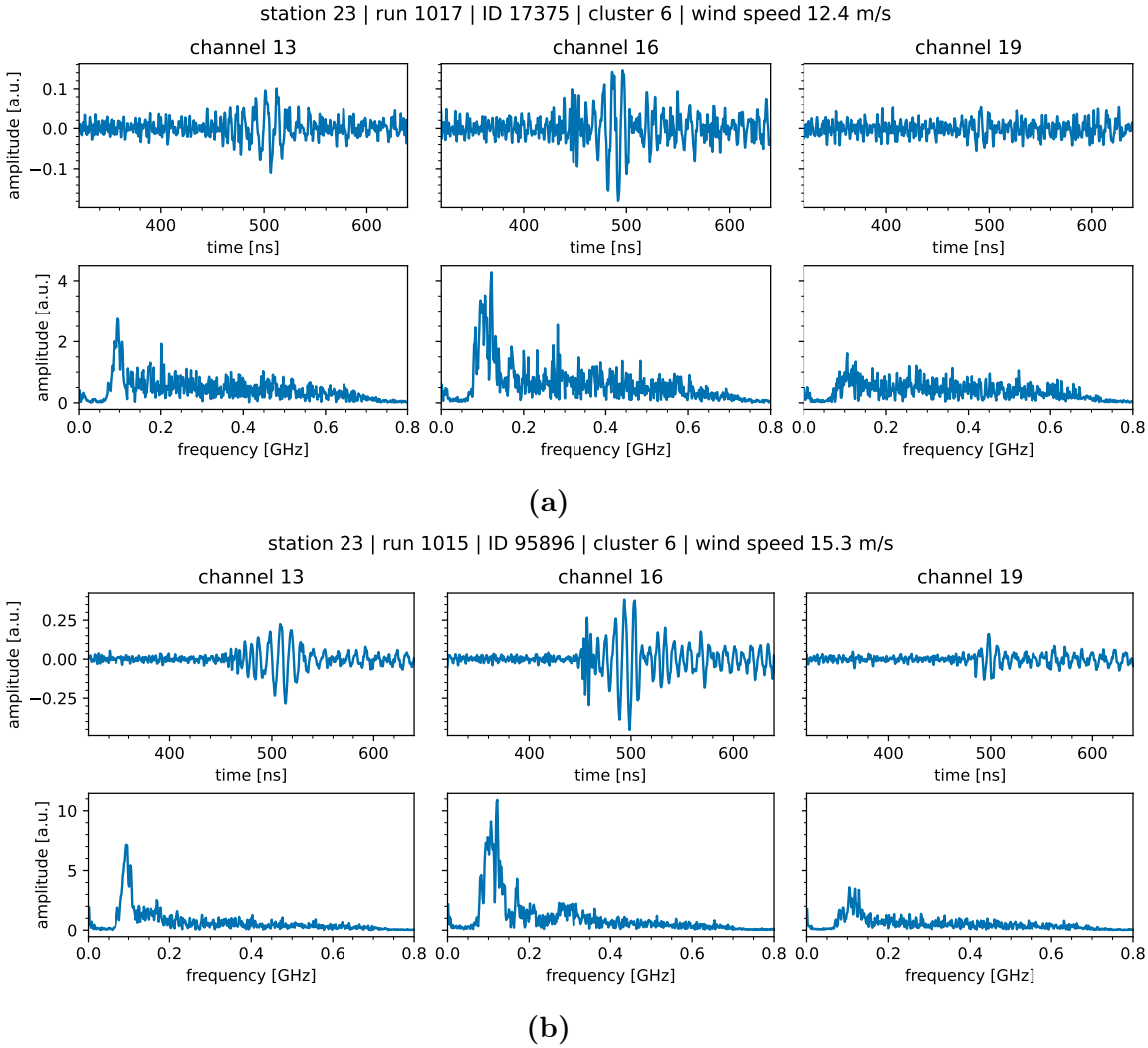
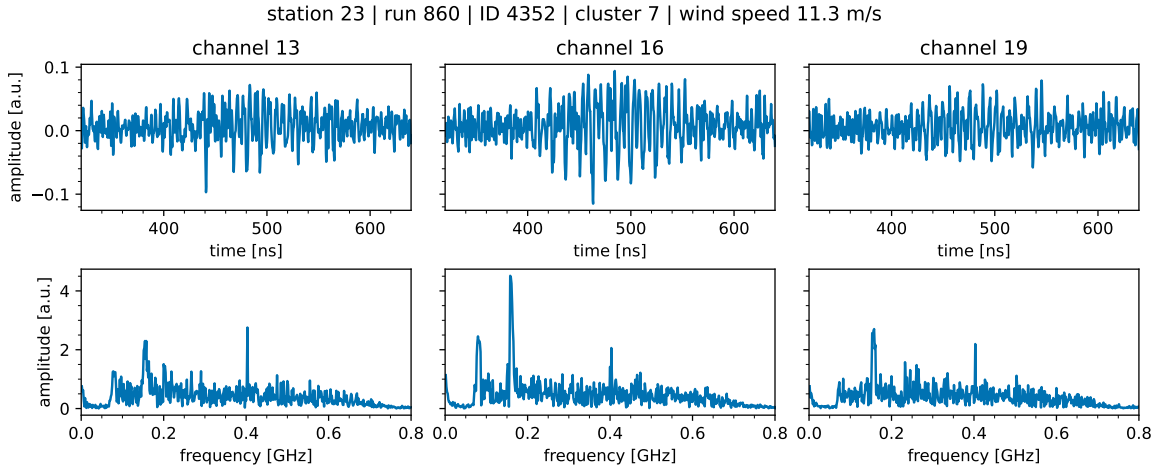
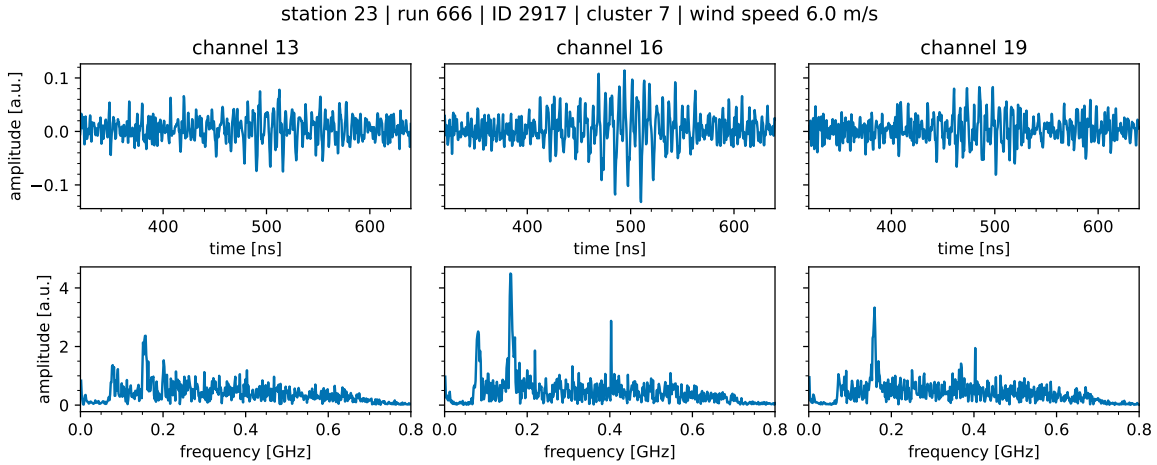


Figure A.25.: Example waveforms and frequency spectra for cluster 6 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.



(a)



(b)

Figure A.26.: Example waveforms and frequency spectra for cluster 7 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

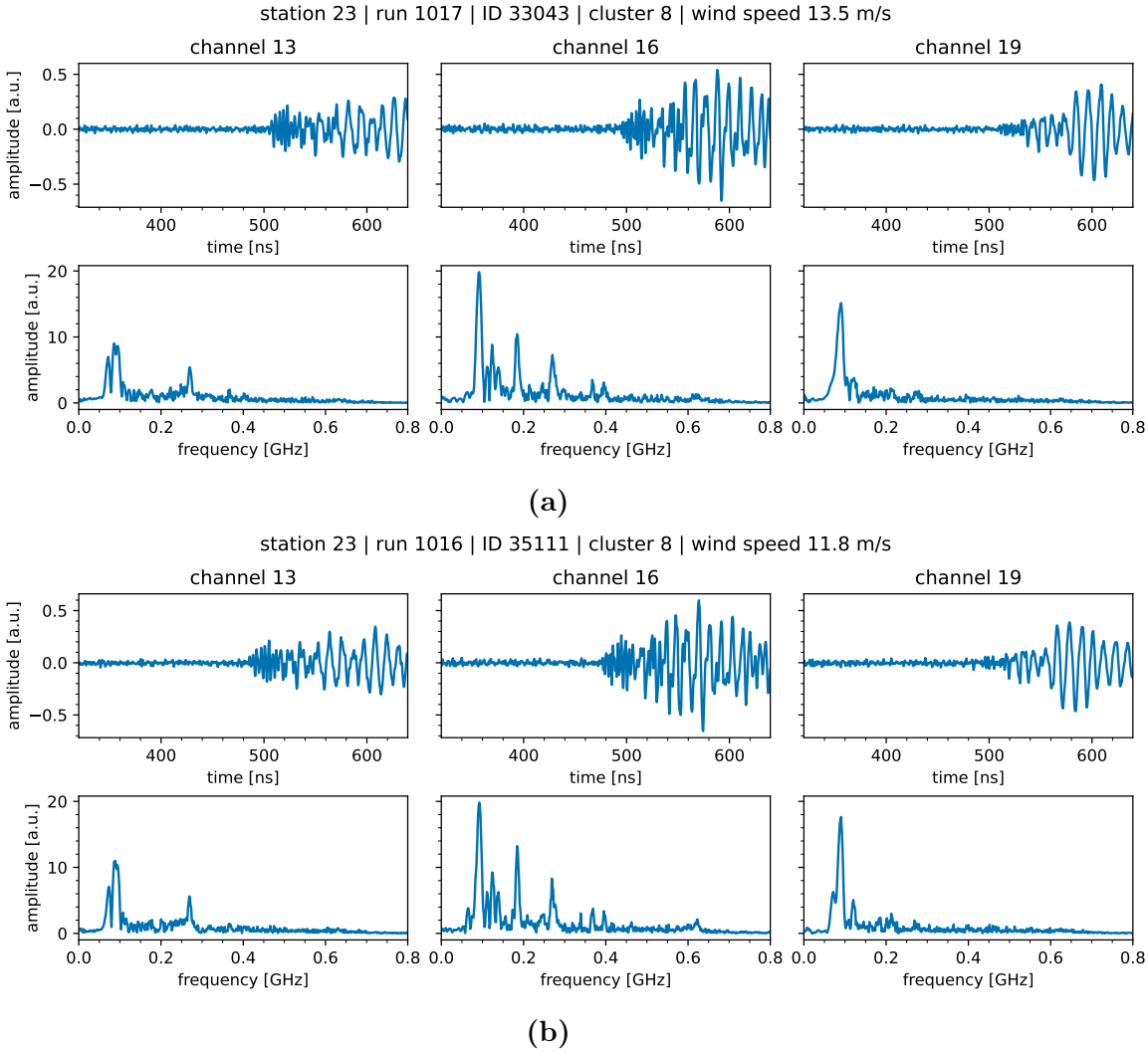


Figure A.27.: Example waveforms and frequency spectra for cluster 8 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

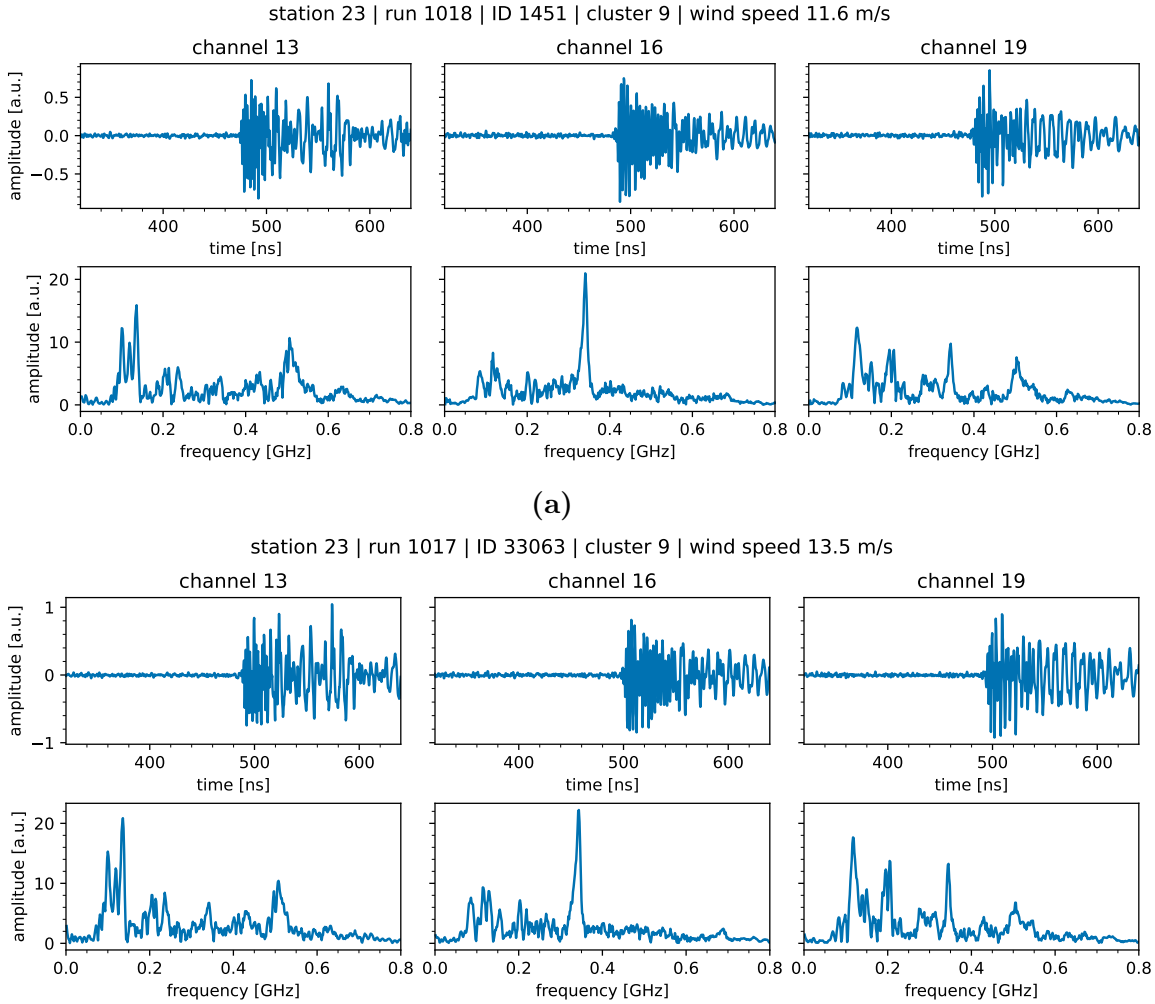


Figure A.28.: Example waveforms and frequency spectra for cluster 9 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

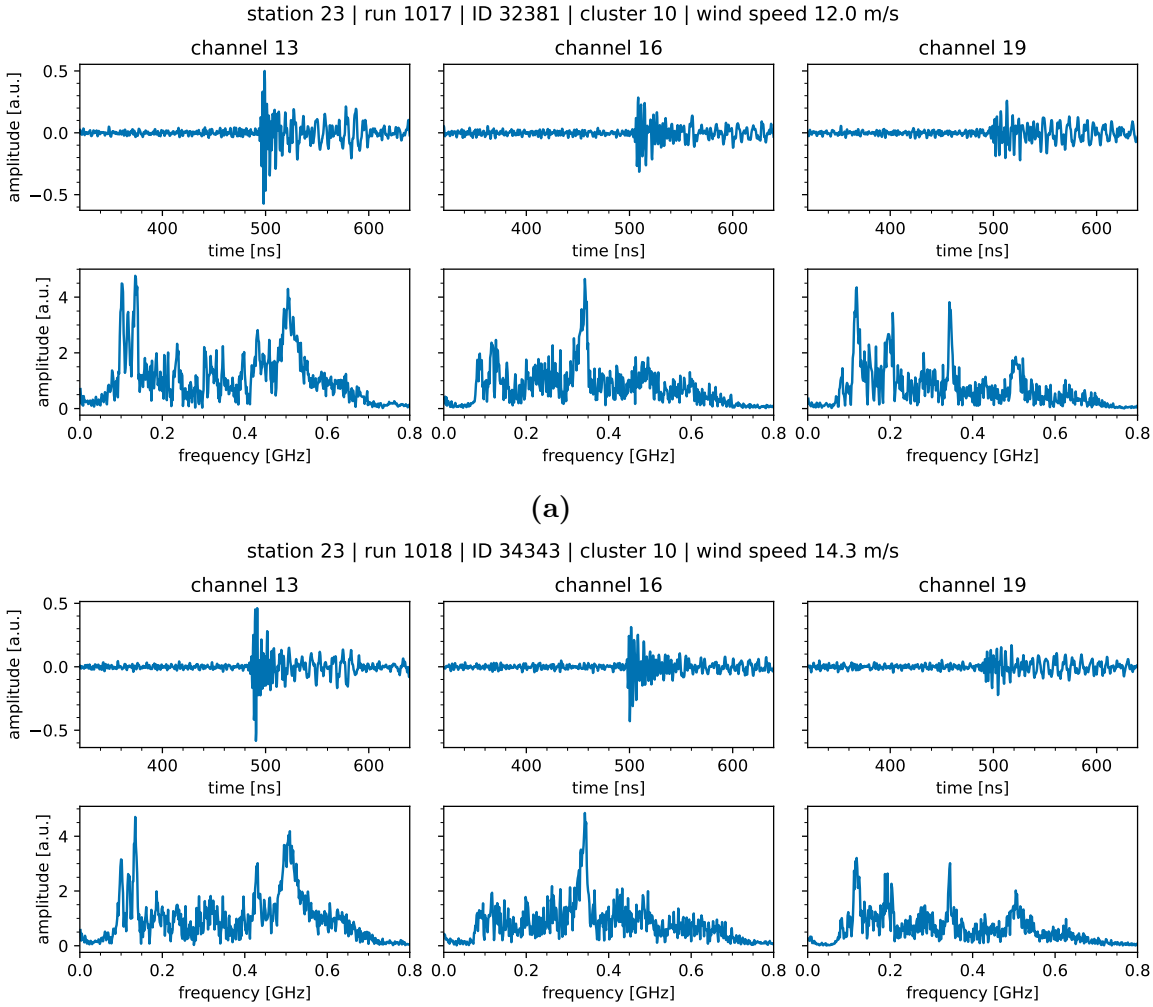


Figure A.29.: Example waveforms and frequency spectra for cluster 10 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

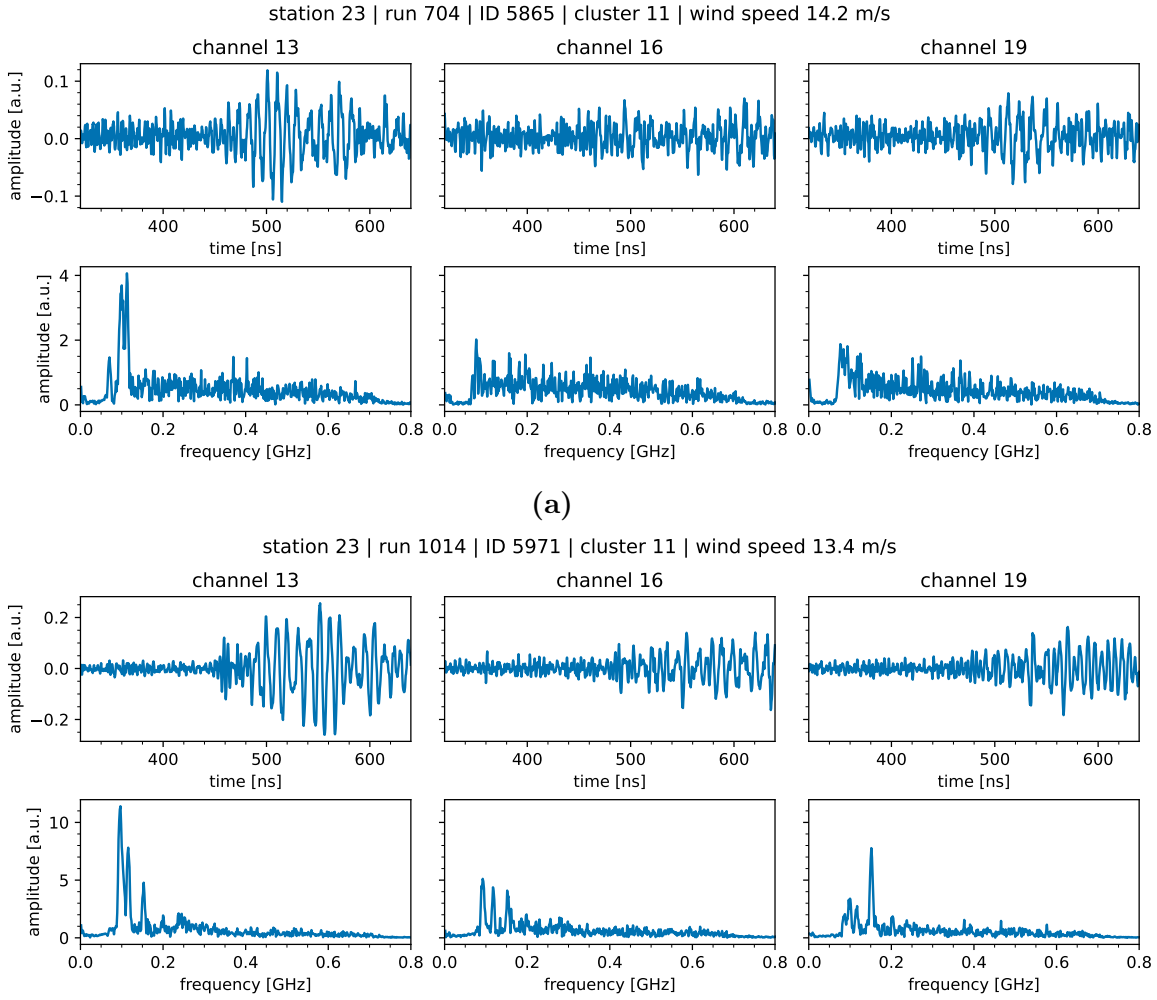


Figure A.30.: Example waveforms and frequency spectra for cluster 11 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

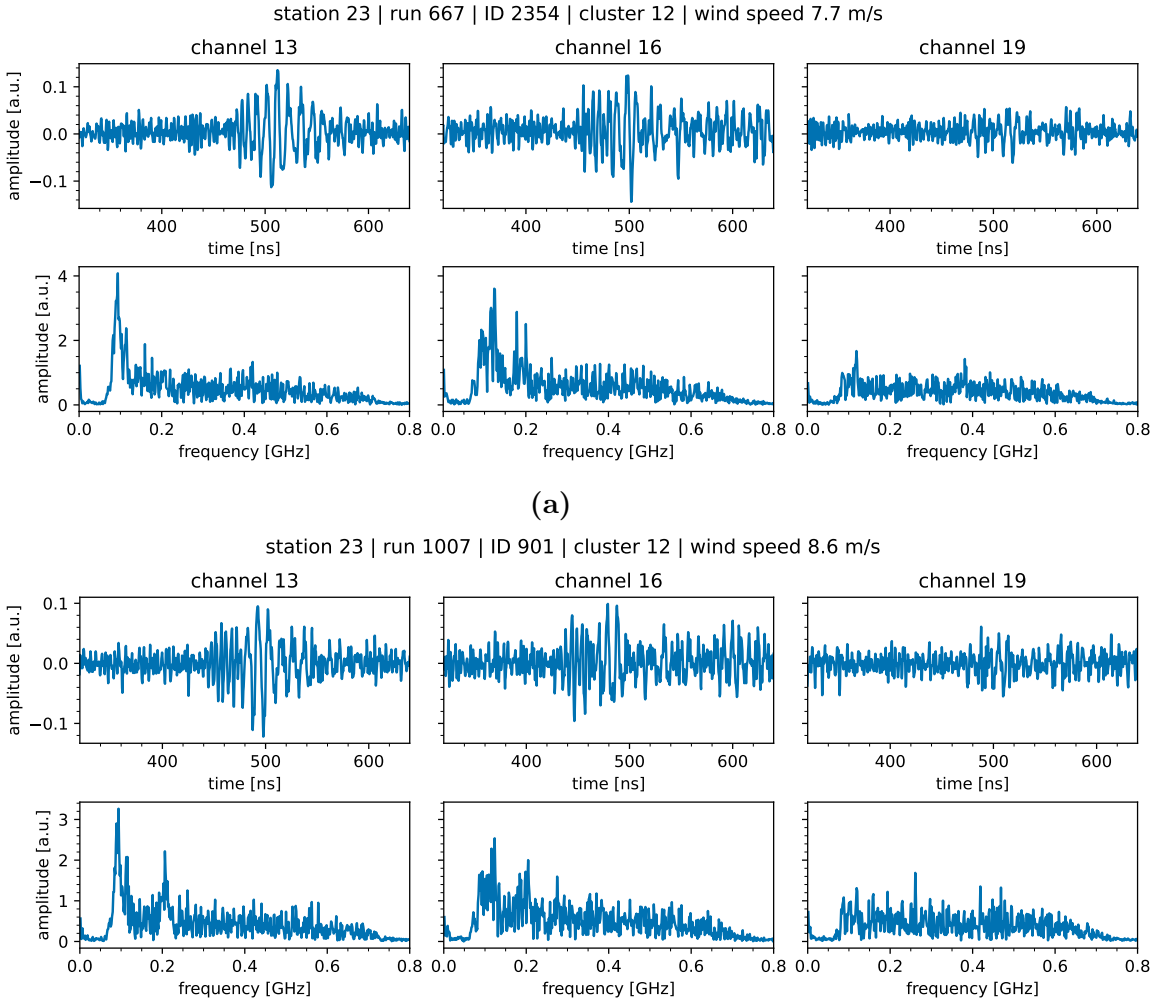
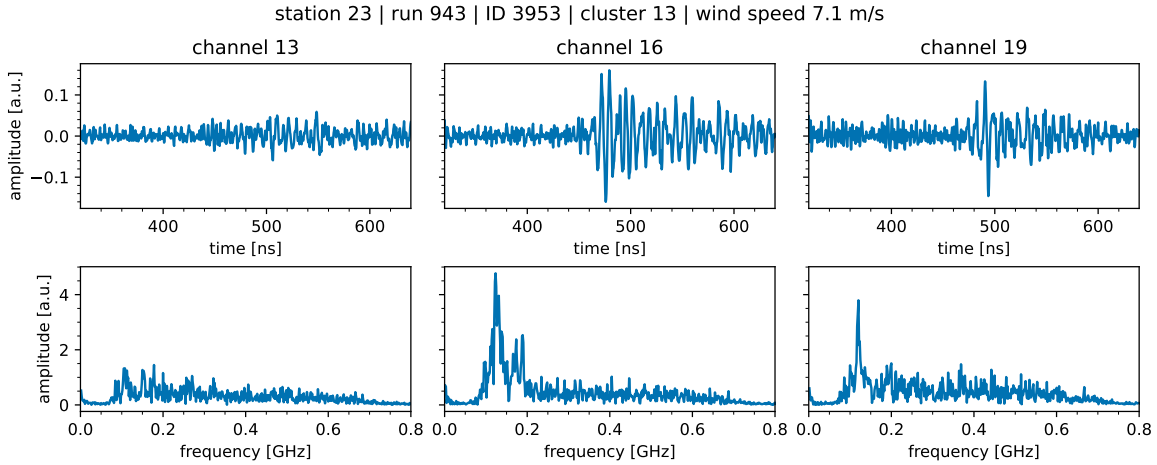
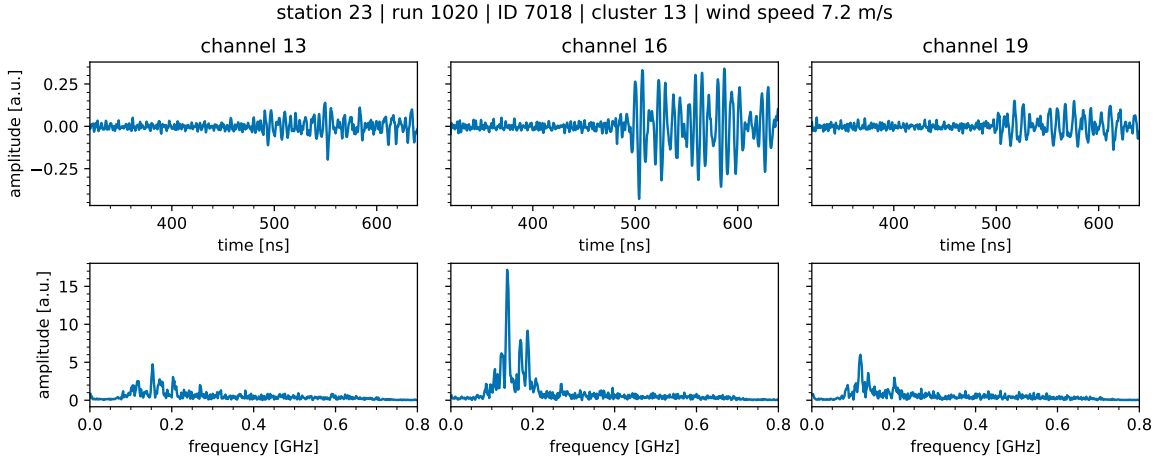


Figure A.31.: Example waveforms and frequency spectra for cluster 12 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

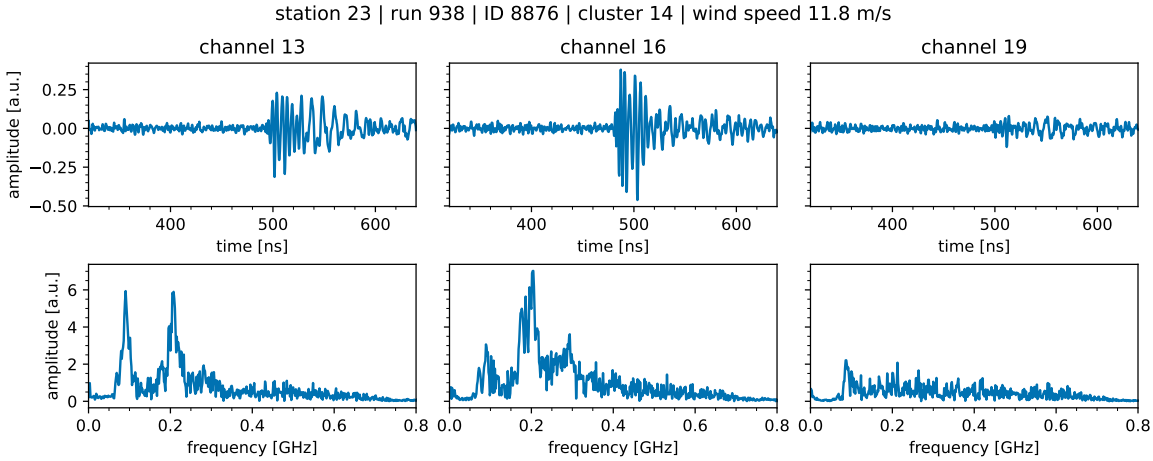


(a)

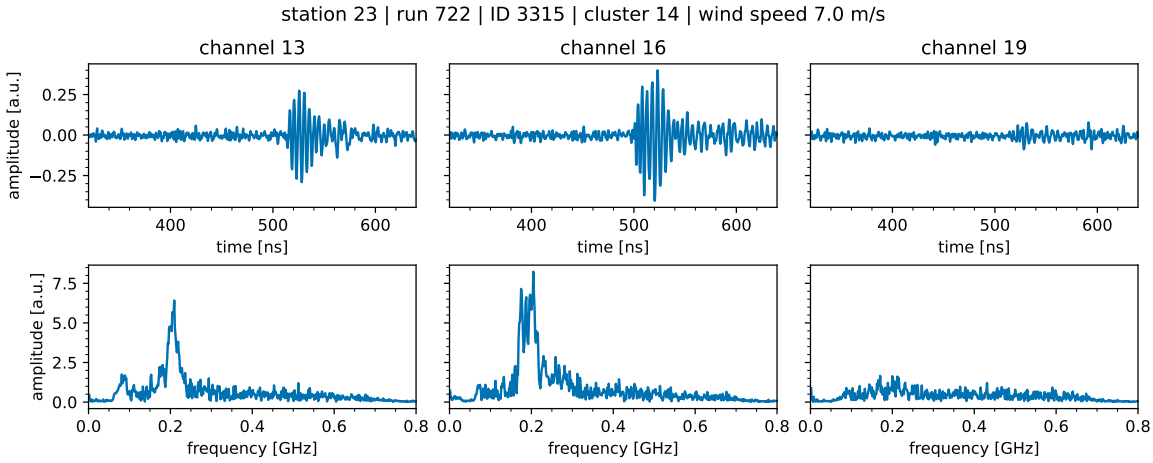


(b)

Figure A.32.: Example waveforms and frequency spectra for cluster 13 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.



(a)



(b)

Figure A.33.: Example waveforms and frequency spectra for cluster 14 (station 23). For better readability, only the second half of the waveforms are shown. The corresponding wind speed is shown in the title of the respective plot.

A.2.2.4. Mean frequency spectra per cluster for different wind speed intervals

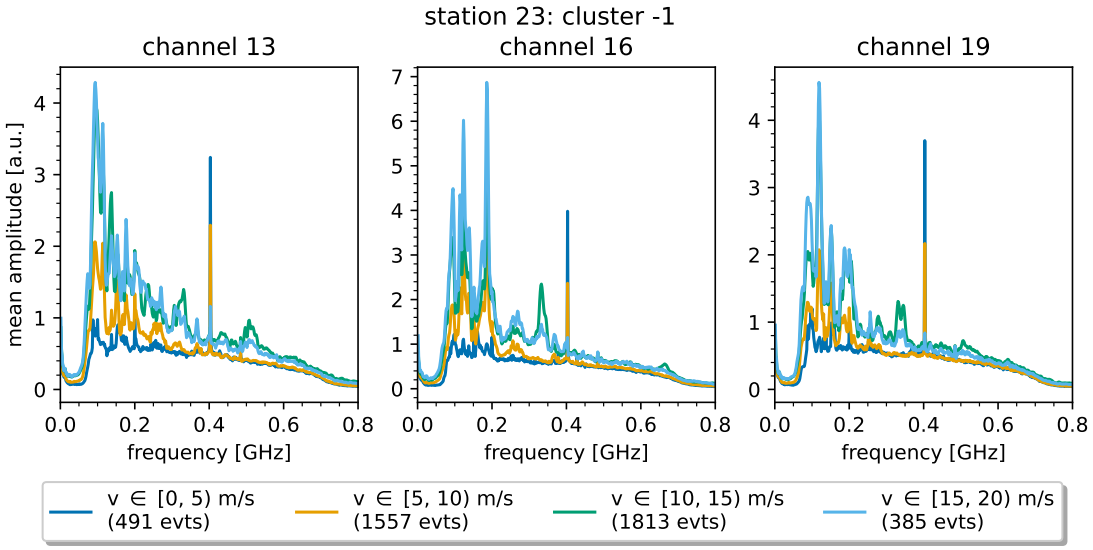


Figure A.34.: Mean frequency spectra per channel and wind speed interval for cluster -1 (station 23).

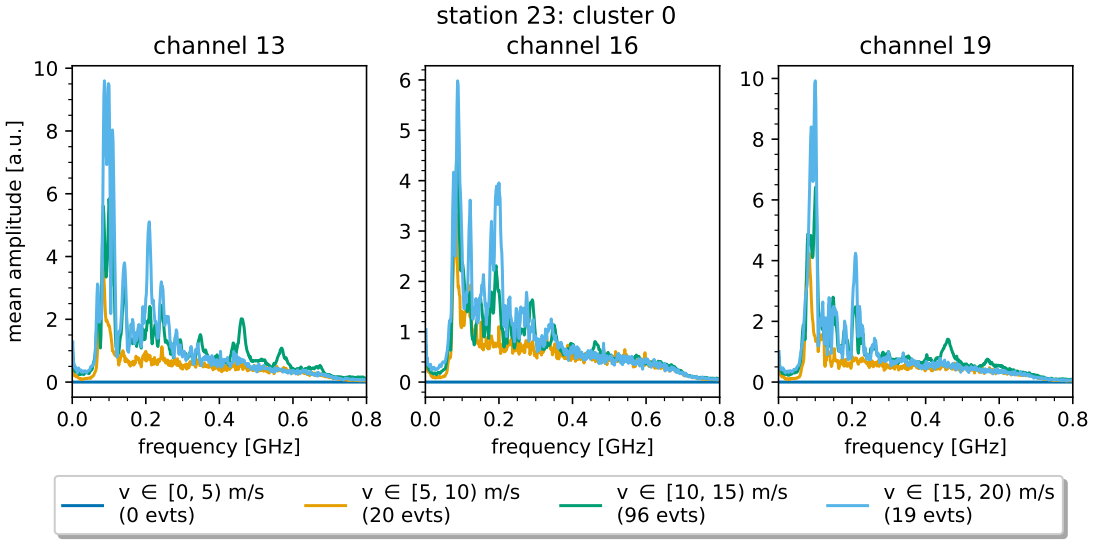


Figure A.35.: Mean frequency spectra per channel and wind speed interval for cluster 0 (station 23).

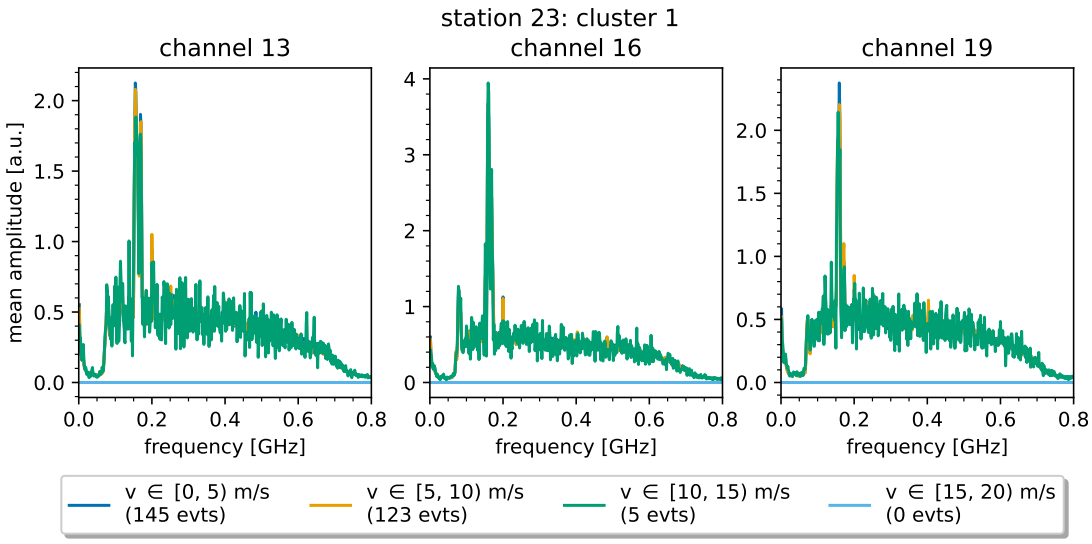


Figure A.36.: Mean frequency spectra per channel and wind speed interval for cluster 1 (station 23).

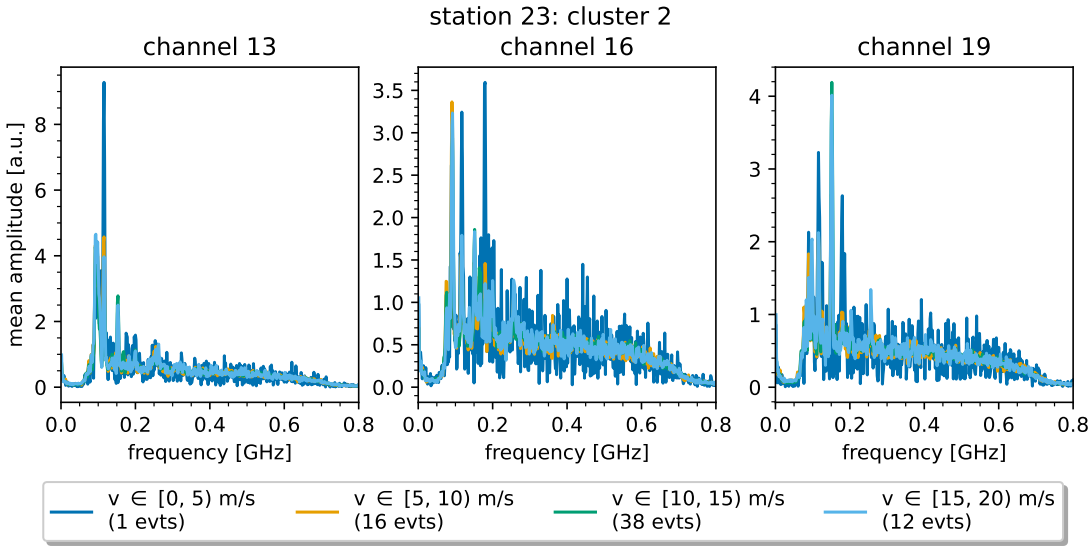


Figure A.37.: Mean frequency spectra per channel and wind speed interval for cluster 2 (station 23).

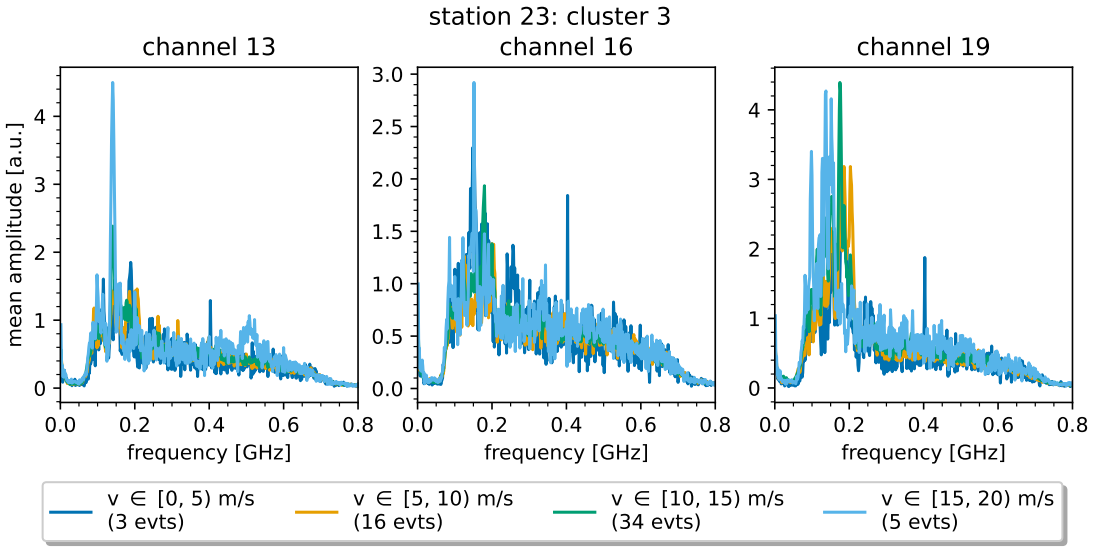


Figure A.38.: Mean frequency spectra per channel and wind speed interval for cluster 3 (station 23).

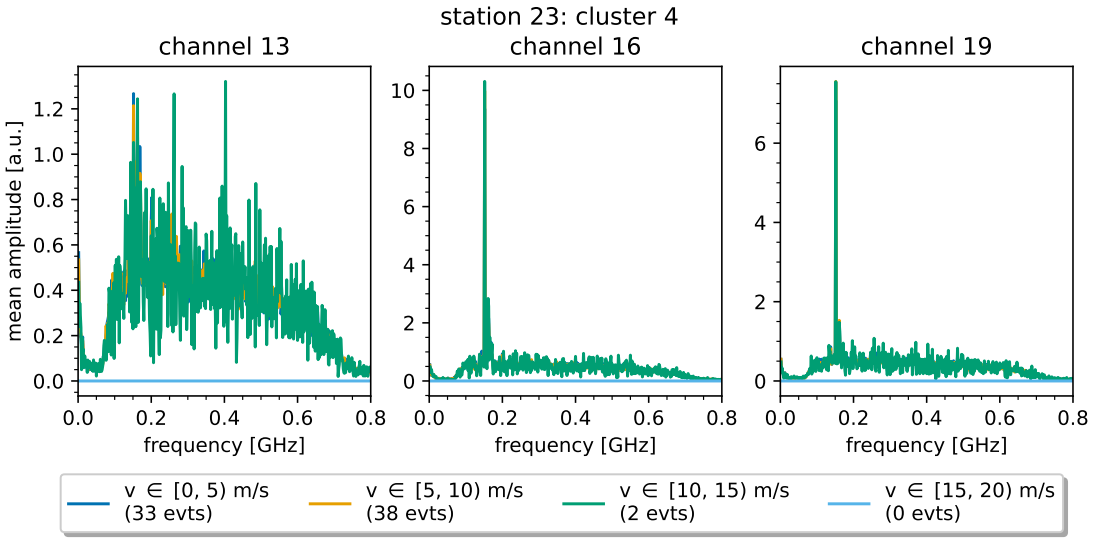


Figure A.39.: Mean frequency spectra per channel and wind speed interval for cluster 4 (station 23).

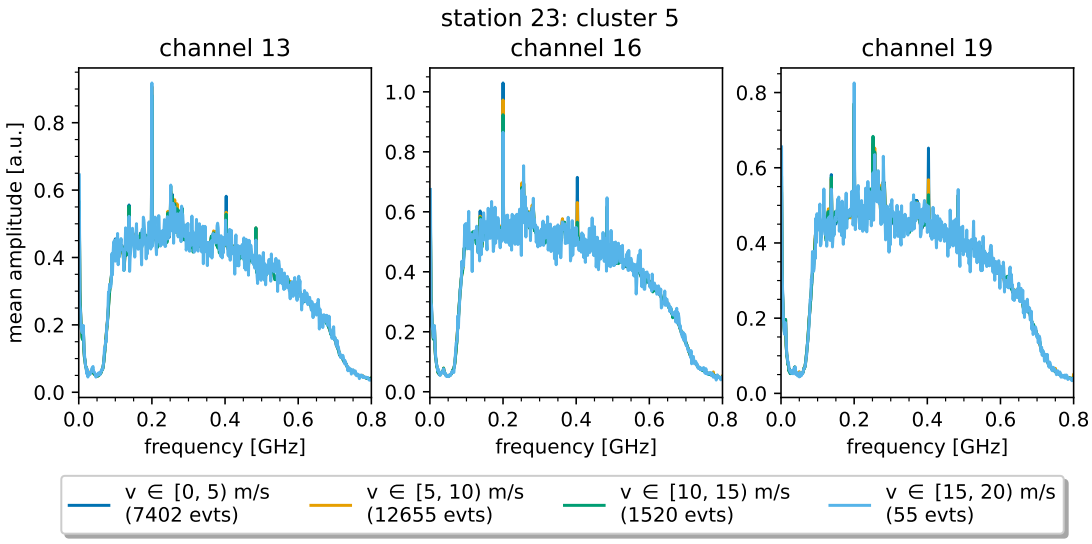


Figure A.40.: Mean frequency spectra per channel and wind speed interval for cluster 5 (station 23).

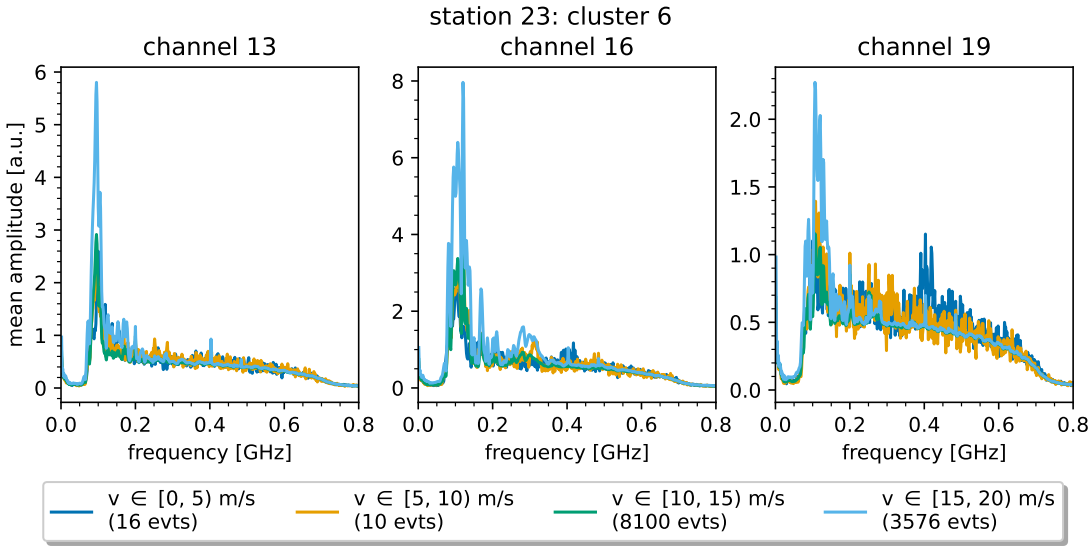


Figure A.41.: Mean frequency spectra per channel and wind speed interval for cluster 6 (station 23).

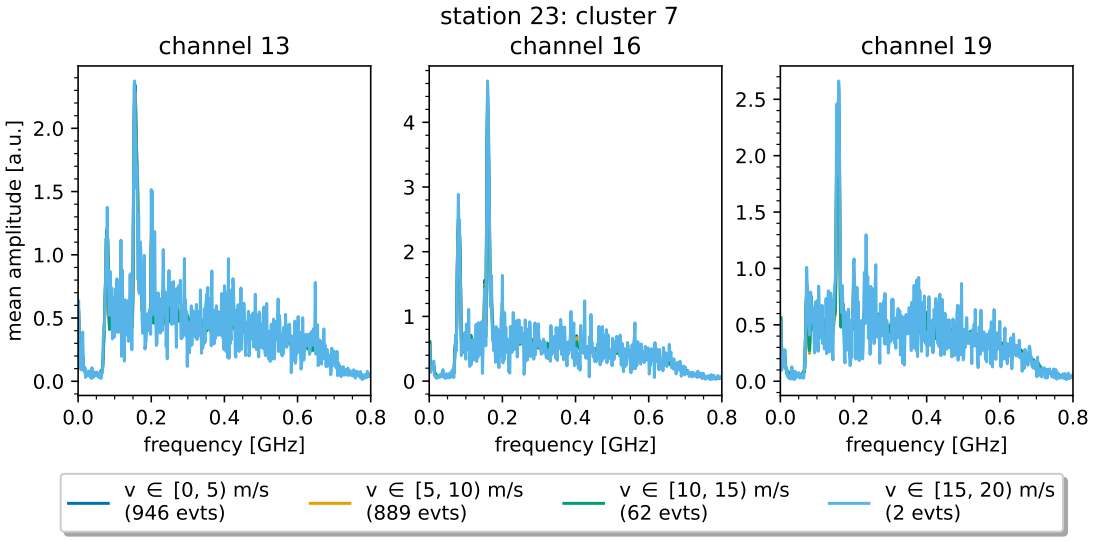


Figure A.42.: Mean frequency spectra per channel and wind speed interval for cluster 7 (station 23).

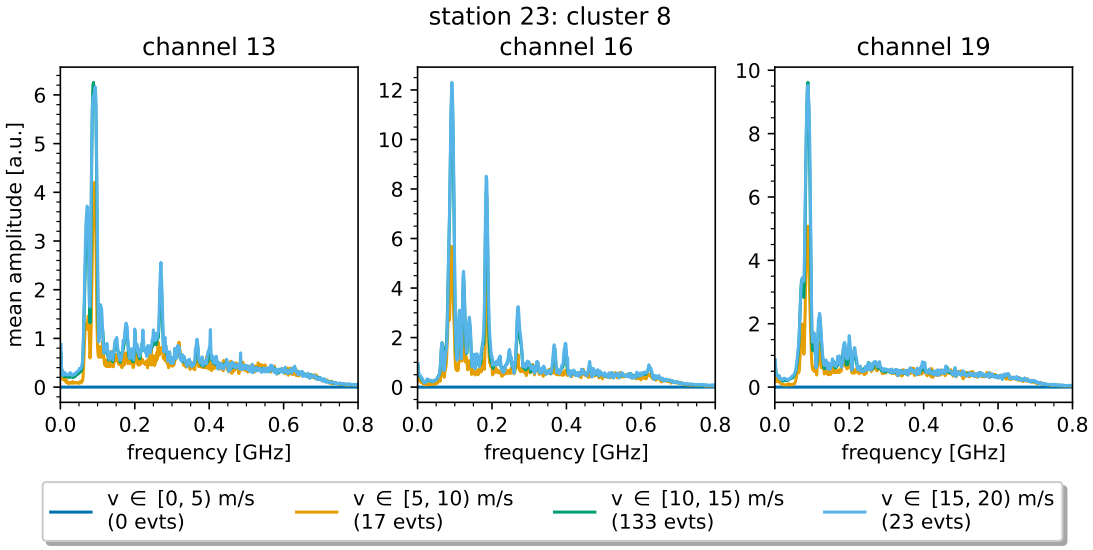


Figure A.43.: Mean frequency spectra per channel and wind speed interval for cluster 8 (station 23).

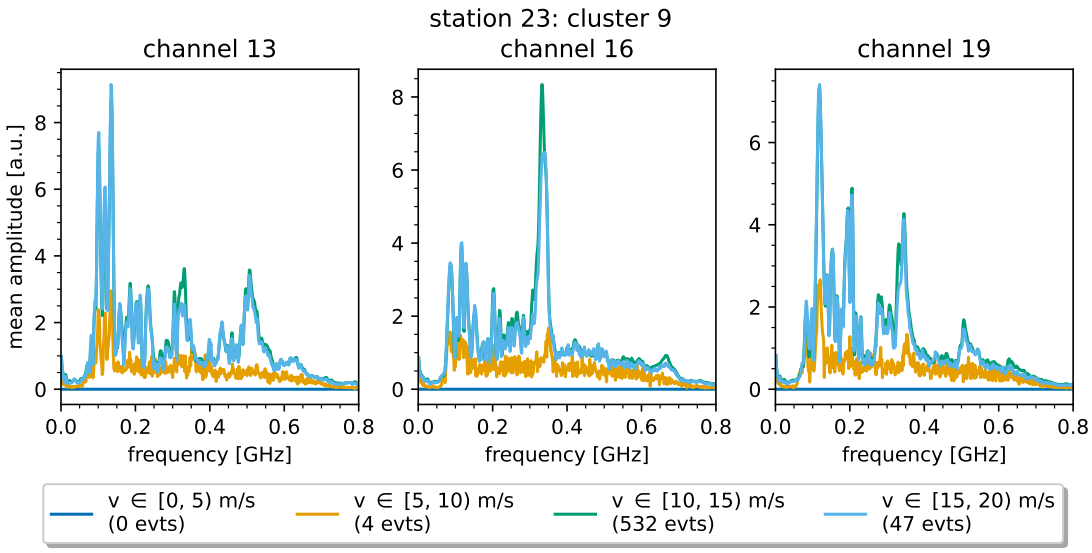


Figure A.44.: Mean frequency spectra per channel and wind speed interval for cluster 9 (station 23).

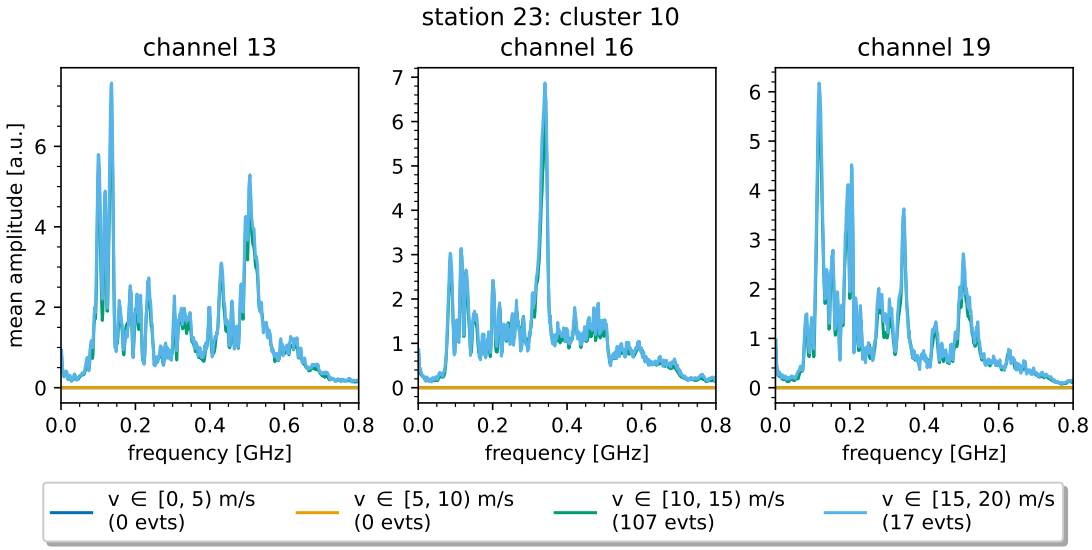


Figure A.45.: Mean frequency spectra per channel and wind speed interval for cluster 10 (station 23).

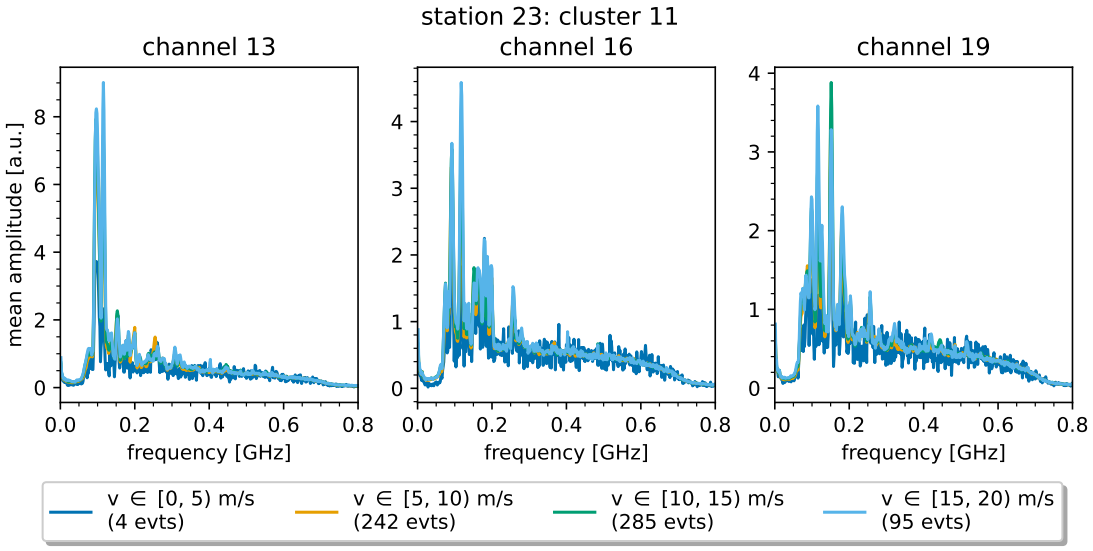


Figure A.46.: Mean frequency spectra per channel and wind speed interval for cluster 11 (station 23).

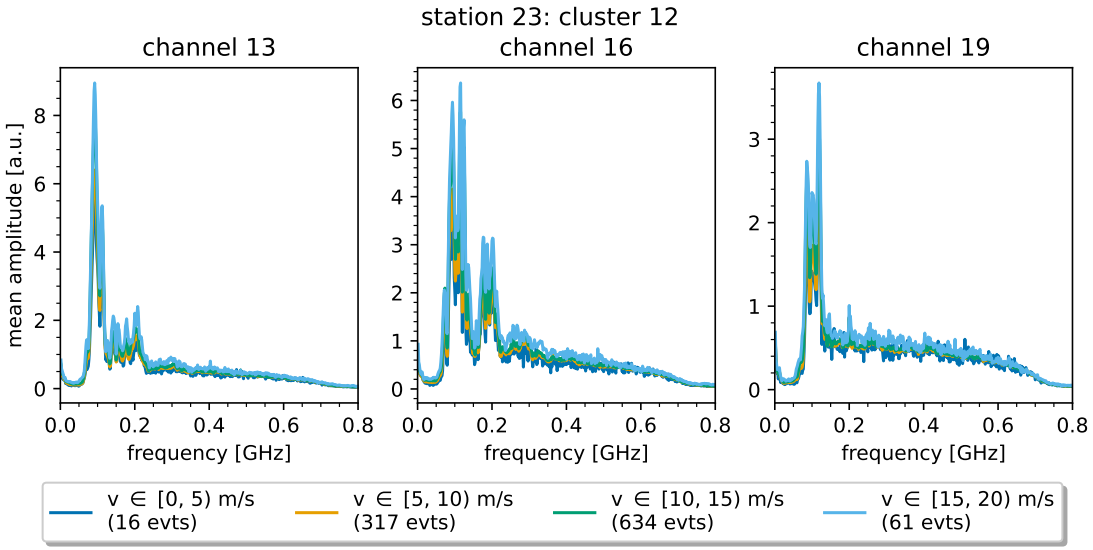


Figure A.47.: Mean frequency spectra per channel and wind speed interval for cluster 12 (station 23).

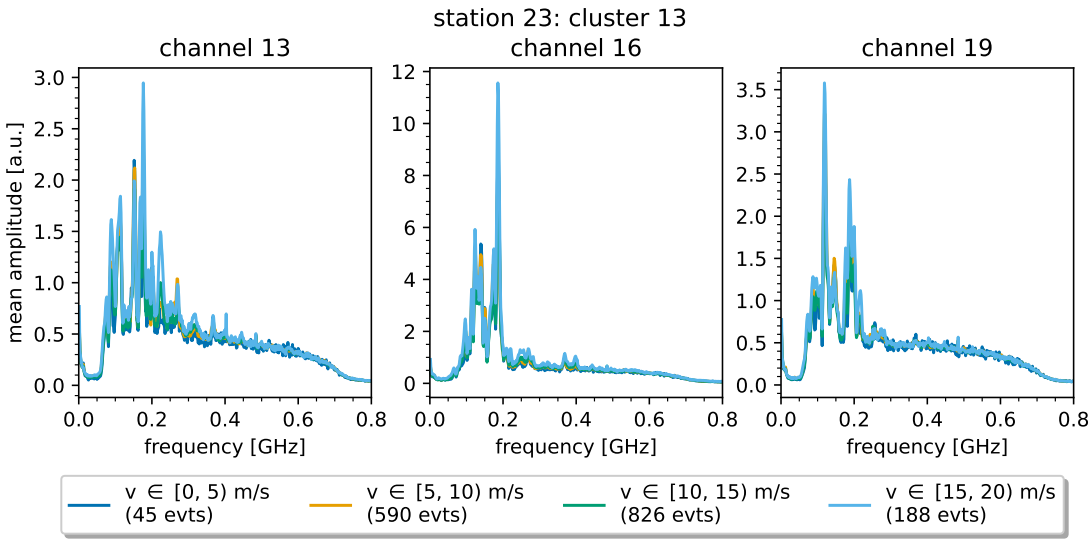


Figure A.48.: Mean frequency spectra per channel and wind speed interval for cluster 13 (station 23).

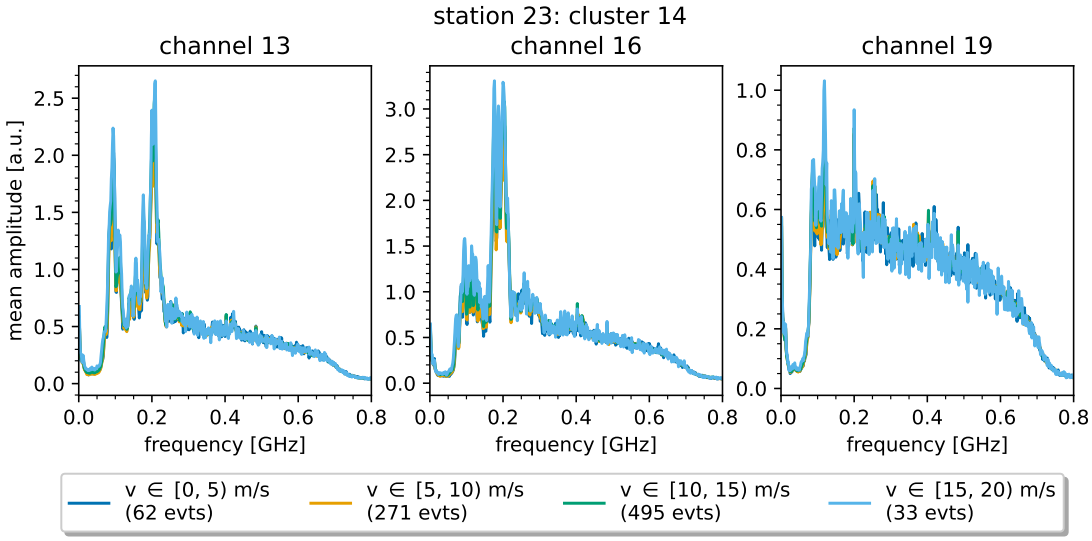


Figure A.49.: Mean frequency spectra per channel and wind speed interval for cluster 14 (station 23).

A.2.2.5. Wind speed over the maximum amplitude of the absolute of the waveform per cluster

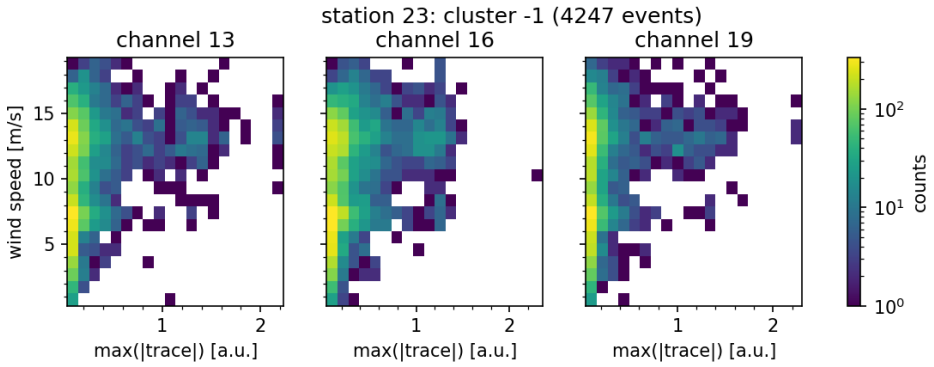


Figure A.50.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster -1 (station 23).

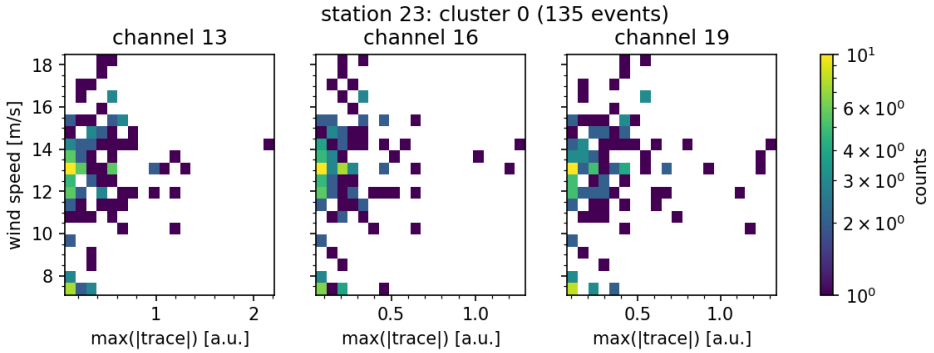


Figure A.51.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 0 (station 23).

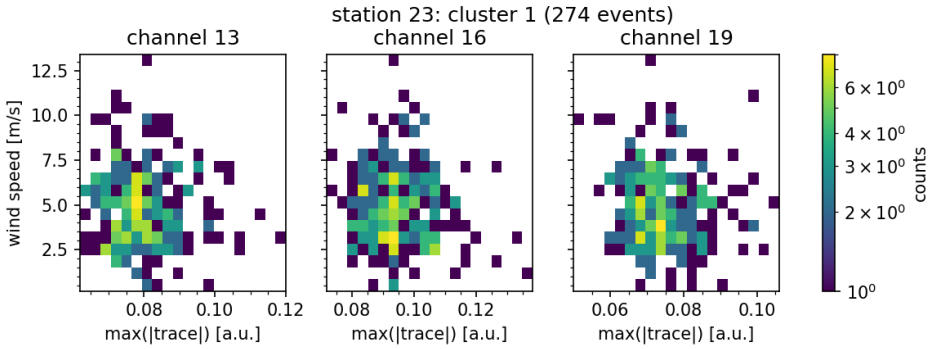


Figure A.52.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 1 (station 23).

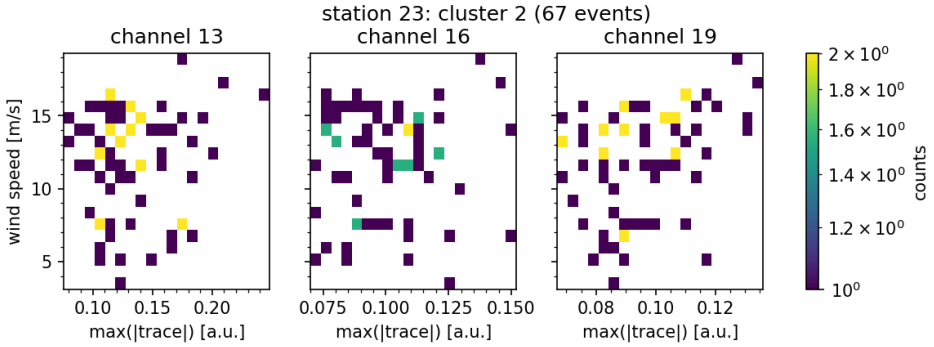


Figure A.53.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 2 (station 23).

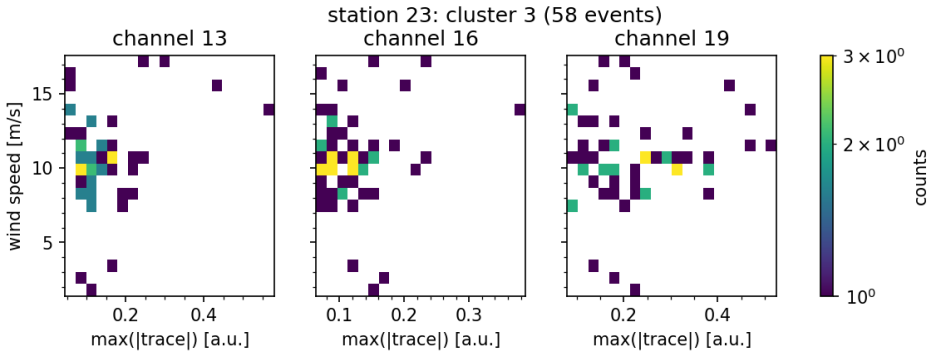


Figure A.54.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 3 (station 23).

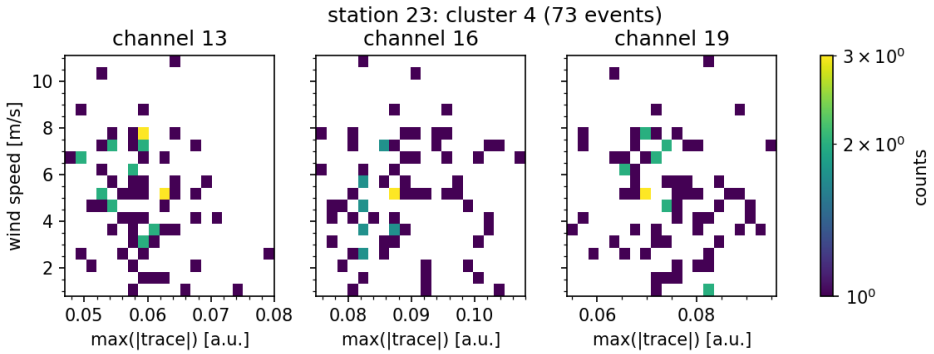


Figure A.55.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 4 (station 23).

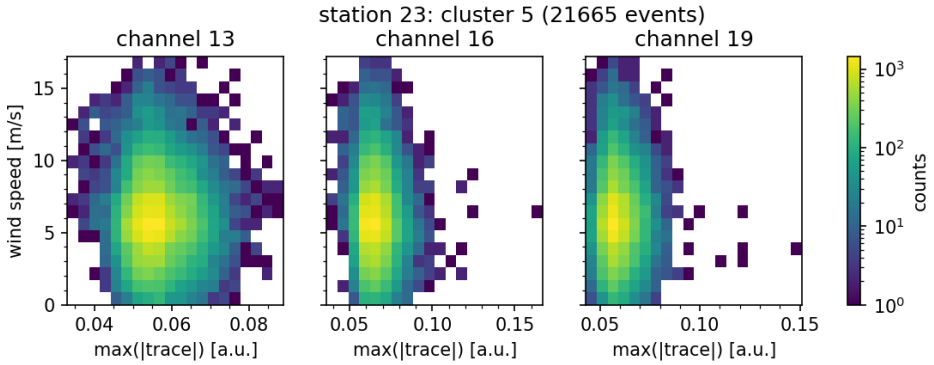


Figure A.56.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 5 (station 23).

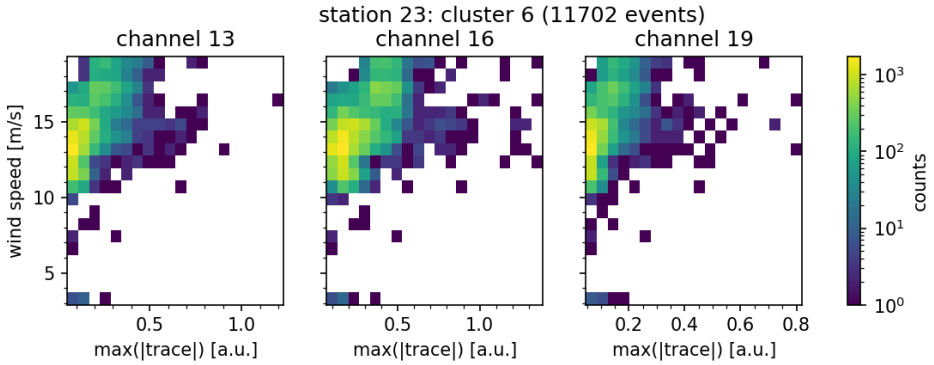


Figure A.57.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 6 (station 23).

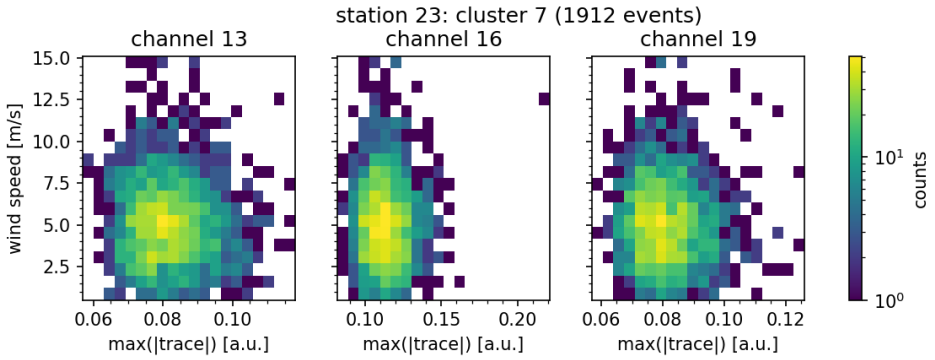


Figure A.58.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 7 (station 23).

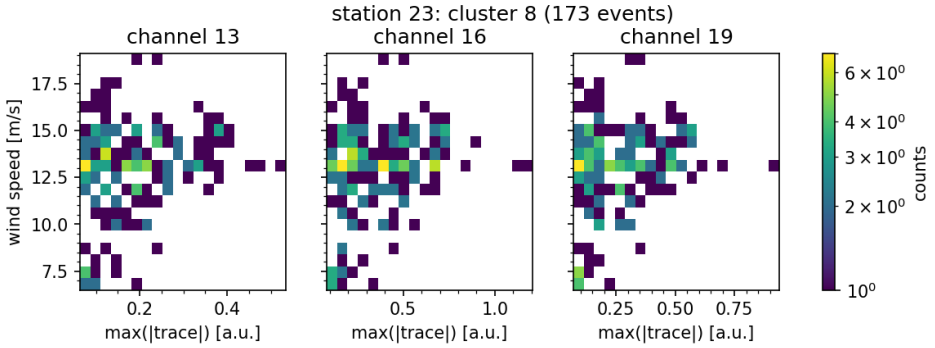


Figure A.59.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 8 (station 23).

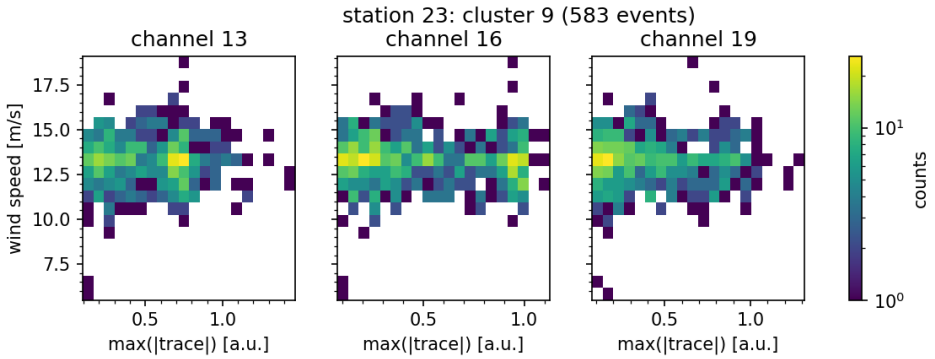


Figure A.60.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 9 (station 23).

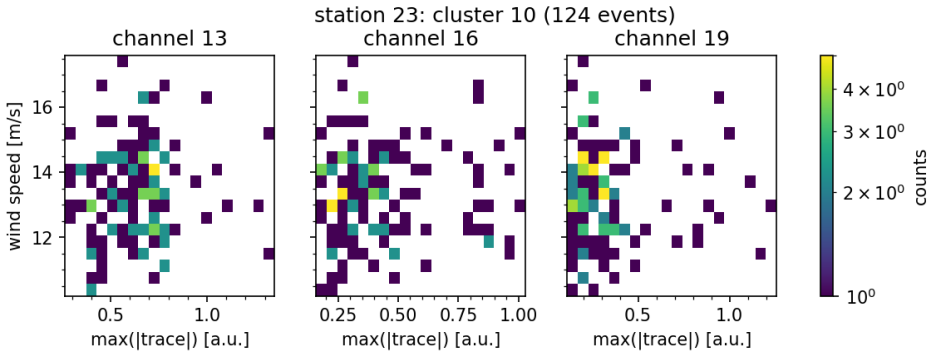


Figure A.61.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 10 (station 23).

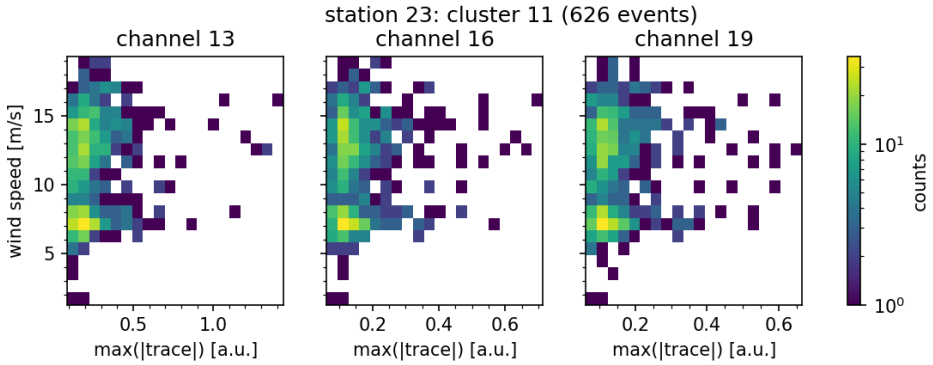


Figure A.62.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 11 (station 23).

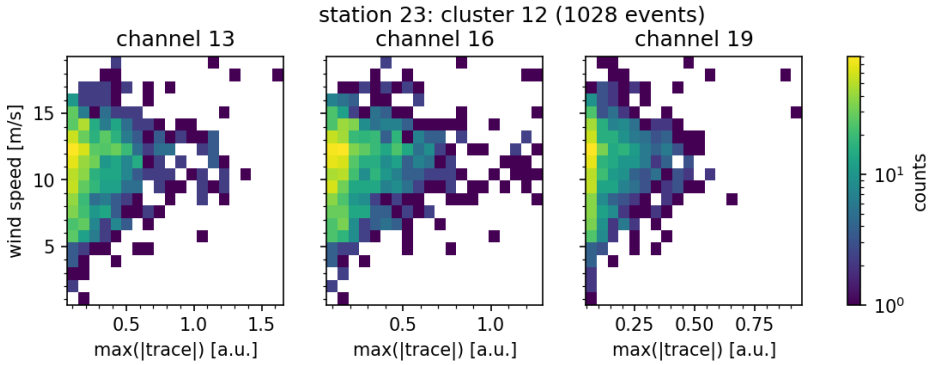


Figure A.63.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 12 (station 23).

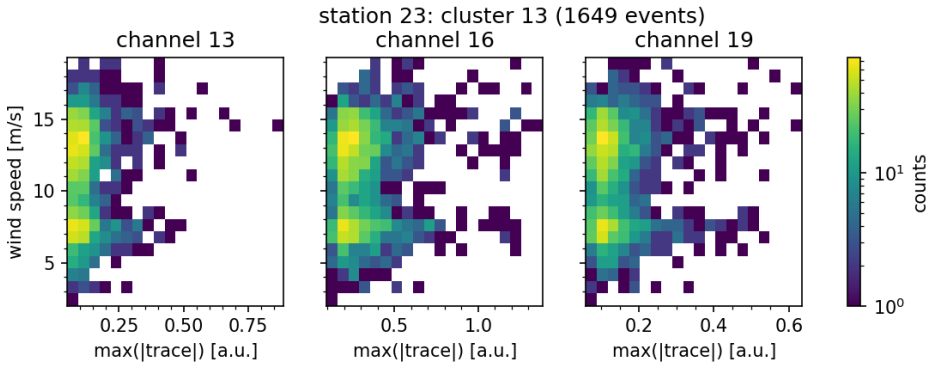


Figure A.64.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 13 (station 23).

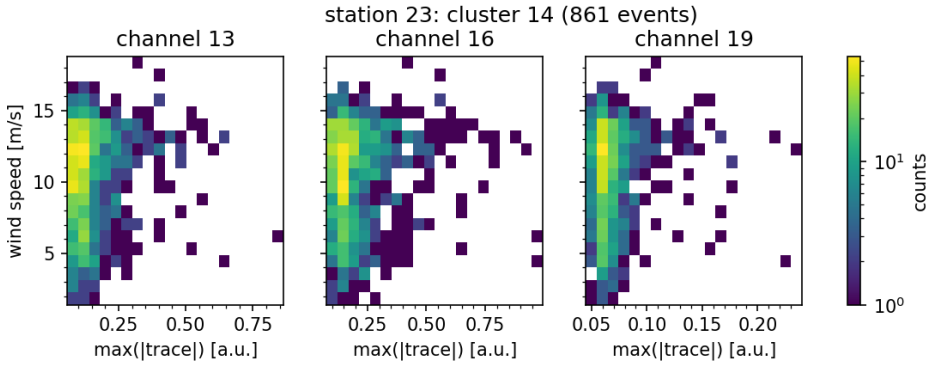


Figure A.65.: 2D histogram of the wind speed over the maximum amplitude of the absolute of the waveform for cluster 14 (station 23).

A.2.3. Clustering results: station 13

parameter	value
`min_samples`	1
`min_cluster_size`	30
`cluster_selection_method`	'leaf'
`cluster_selection_epsilon`	0.495
`leaf_size`	1

Table A.4.: Non-default HDBSCAN parameters used for the final clustering for station 13.

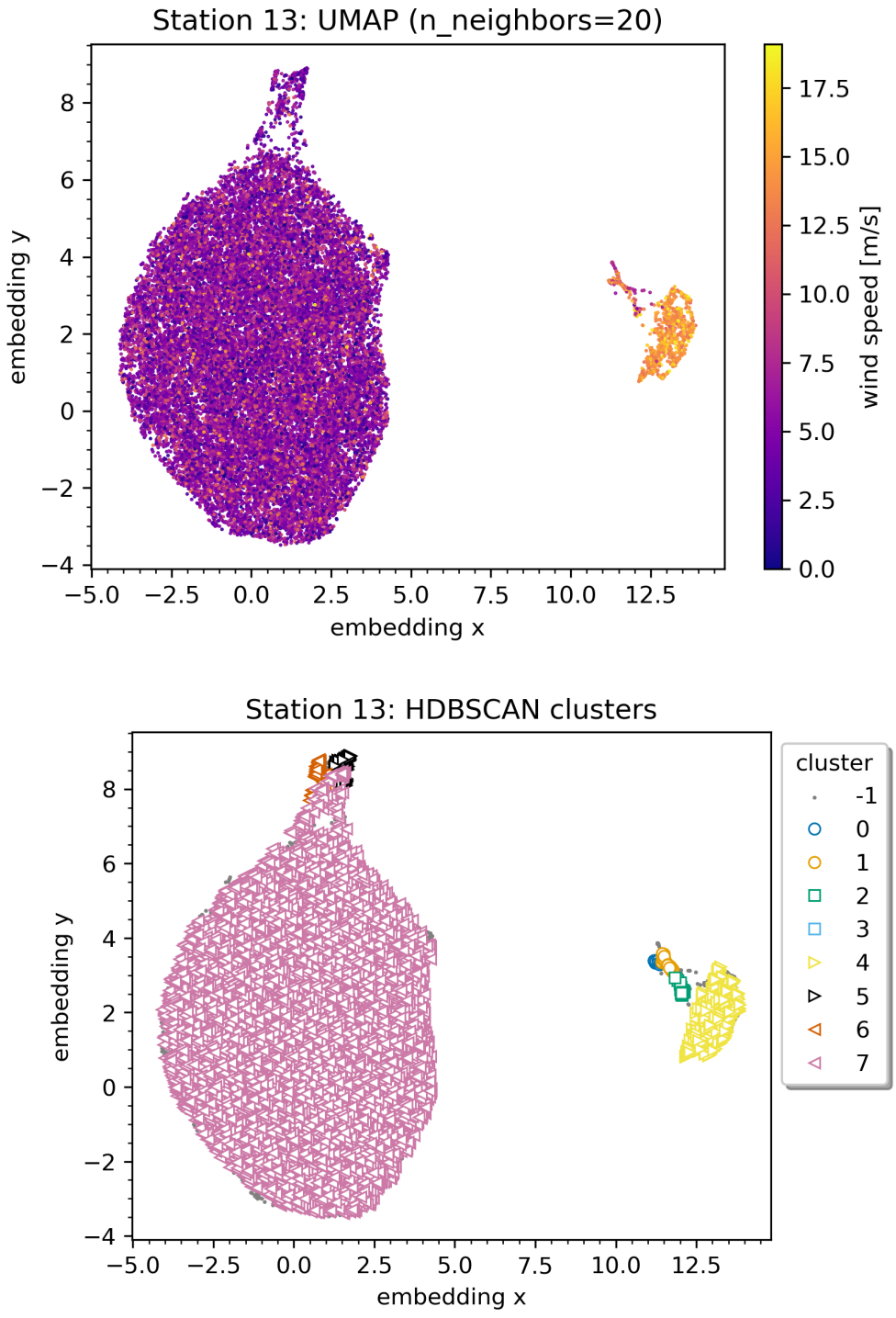


Figure A.66.: UMAP representation of the latent vectors of station 13 shallow- and force-triggered events with the corresponding wind speed encoded by color (top) and overlayed with clusters found by HDBSCAN (bottom). Cluster -1 denotes those events that are not assigned to any cluster.

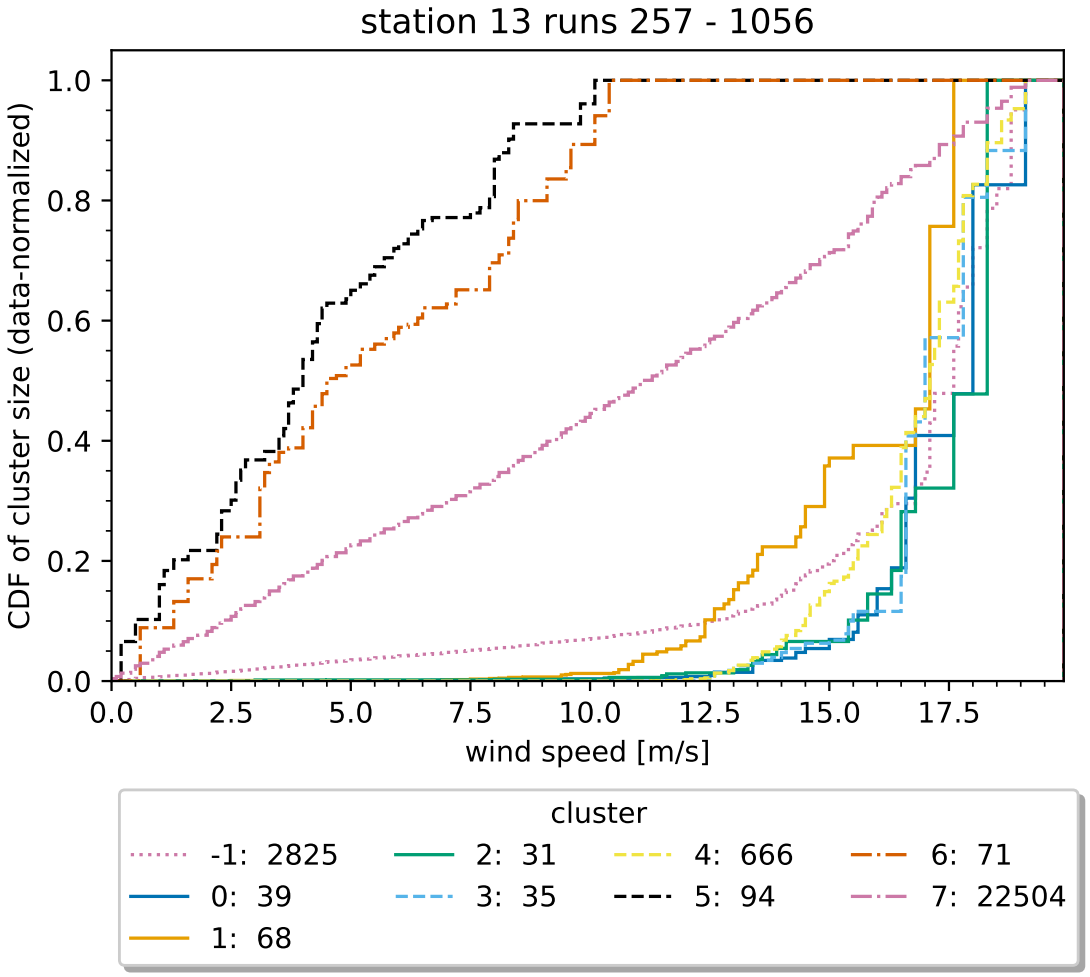


Figure A.67.: Station 13: CDF of the cluster size as a function of the wind speed (data-normalized). The total number of events per cluster is shown in the legend.

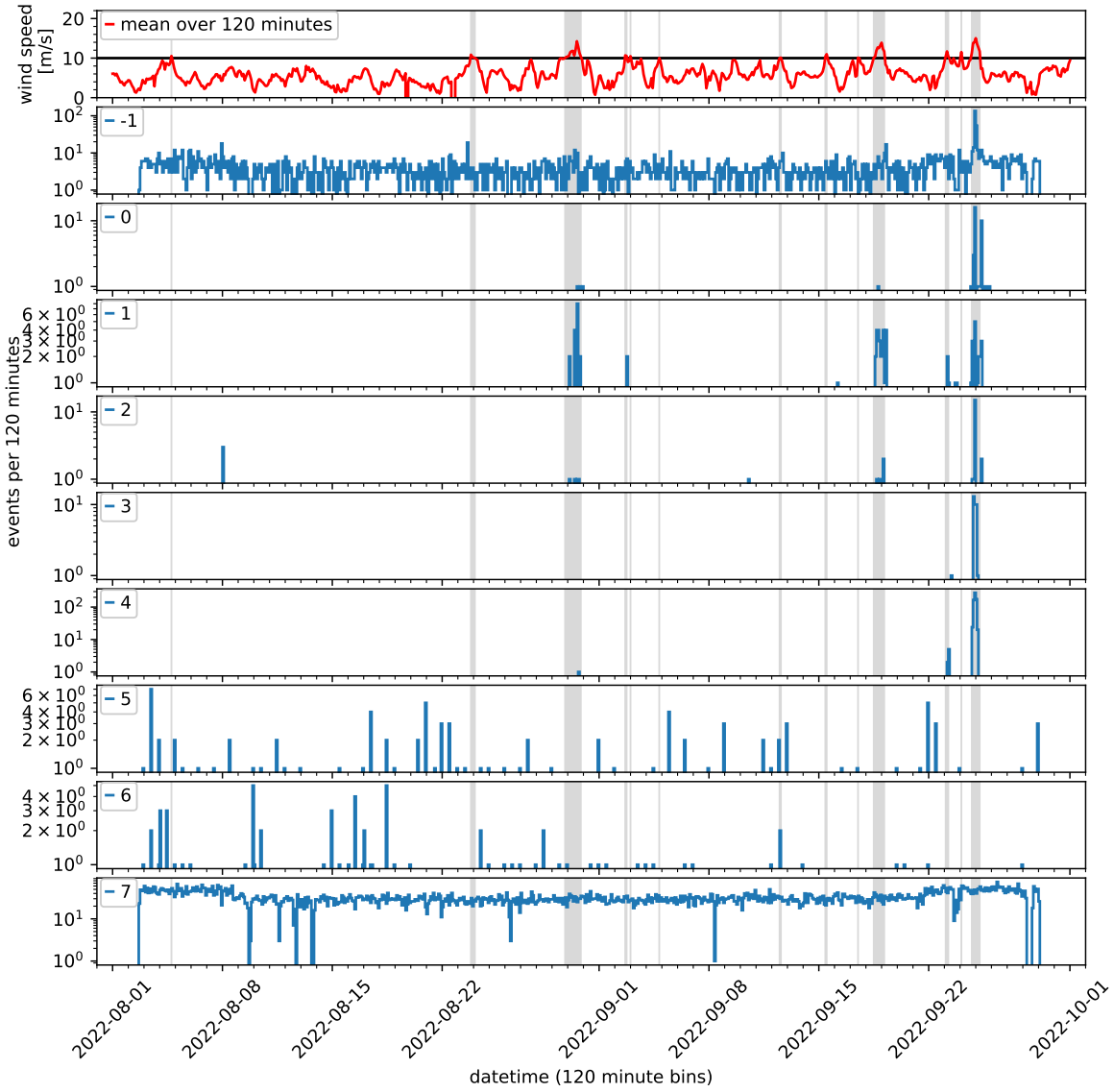
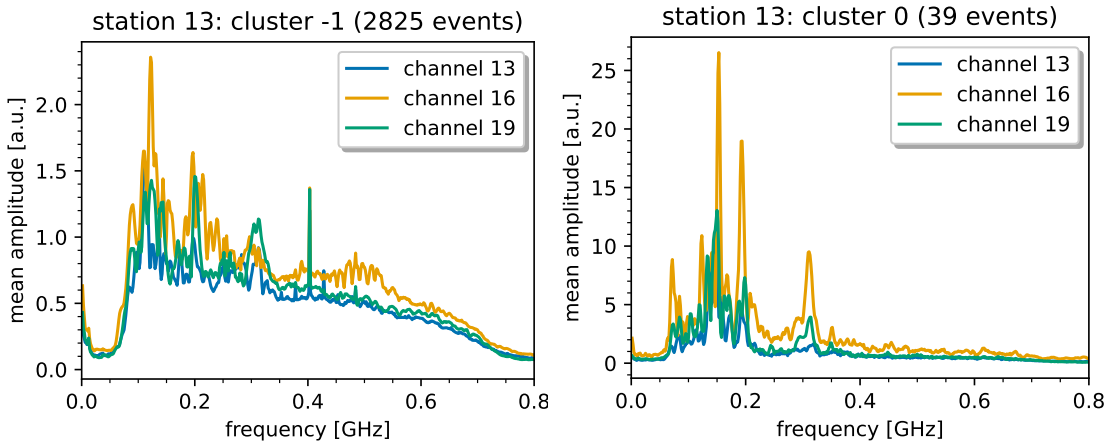


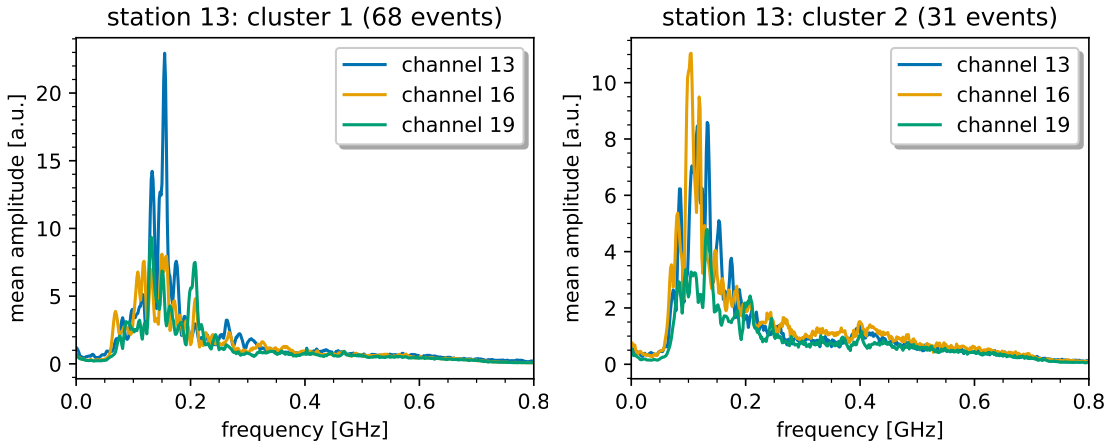
Figure A.68.: Station 13: event rate per cluster found by HDBSCAN as well as wind speed. Time intervals with wind speed $\geq 10\,\mathrm{m/s}$ are shaded in gray. Time is given in UTC.

A.2.3.1. Mean frequency spectra per cluster



(a) (b)

Figure A.69.: Mean frequency spectra per channel for a) cluster −1 and b) cluster 0 (station 13).



(a) (b)

Figure A.70.: Mean frequency spectra per channel for a) cluster 1 and b) cluster 2 (station 13).

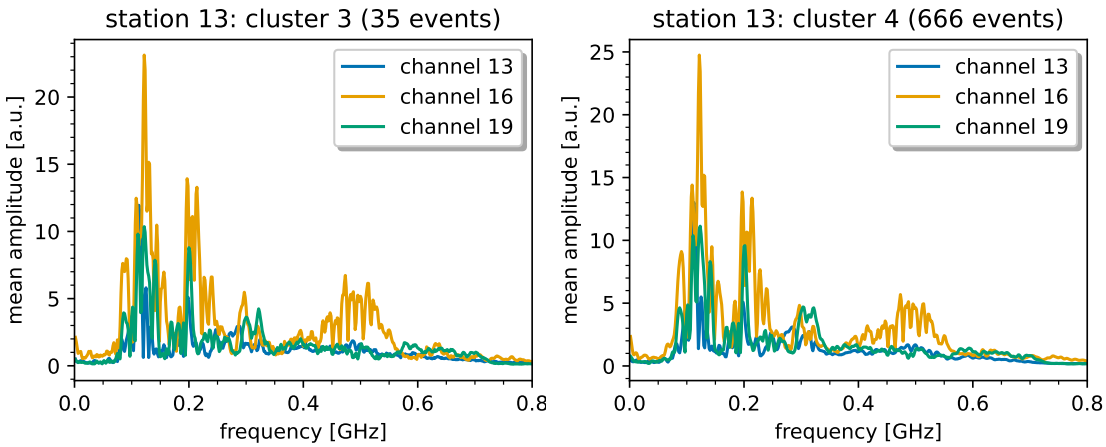


Figure A.71.: Mean frequency spectra per channel for a) cluster 3 and b) cluster 4 (station 13).

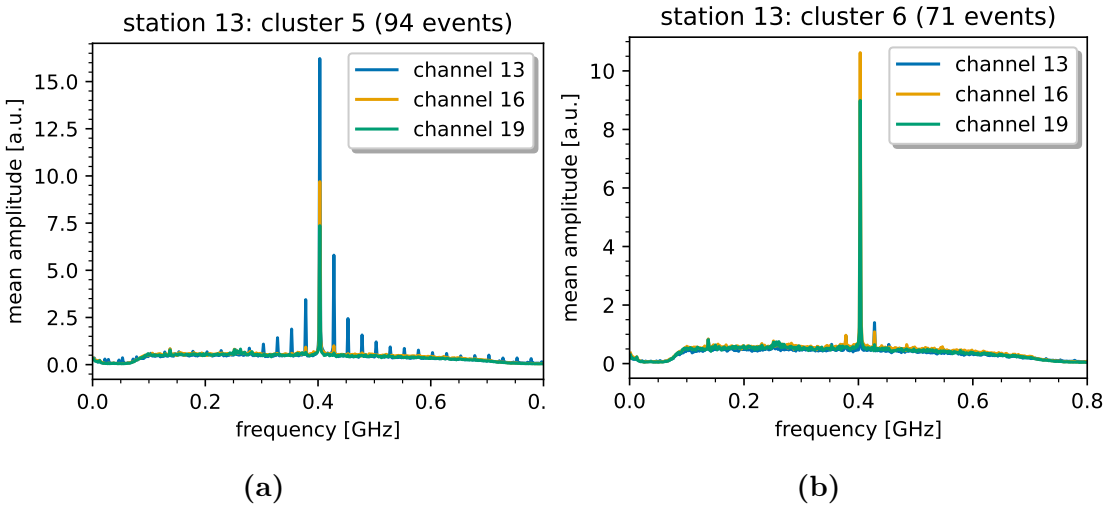


Figure A.72.: Mean frequency spectra per channel for a) cluster 5 and b) cluster 6 (station 13).

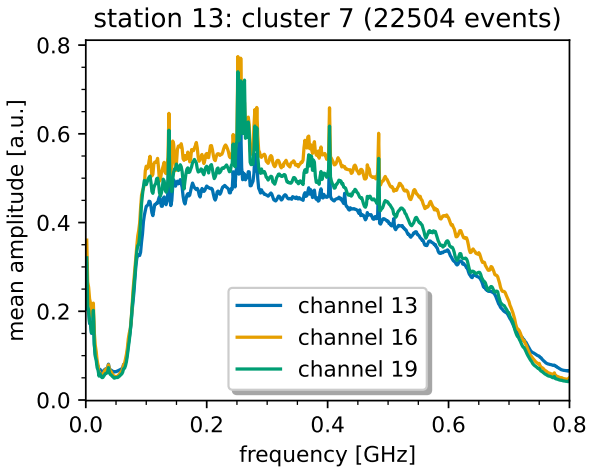


Figure A.73.: Mean frequency spectra per channel for cluster 7 (station 13).

A.2.4. Clustering results: station 24

parameter	value
`min_samples`	1
`min_cluster_size`	50
`cluster_selection_method`	'leaf'
`cluster_selection_epsilon`	0.6
`leaf_size`	1

Table A.5.: Non-default HDBSCAN parameters used for the final clustering for station 24.

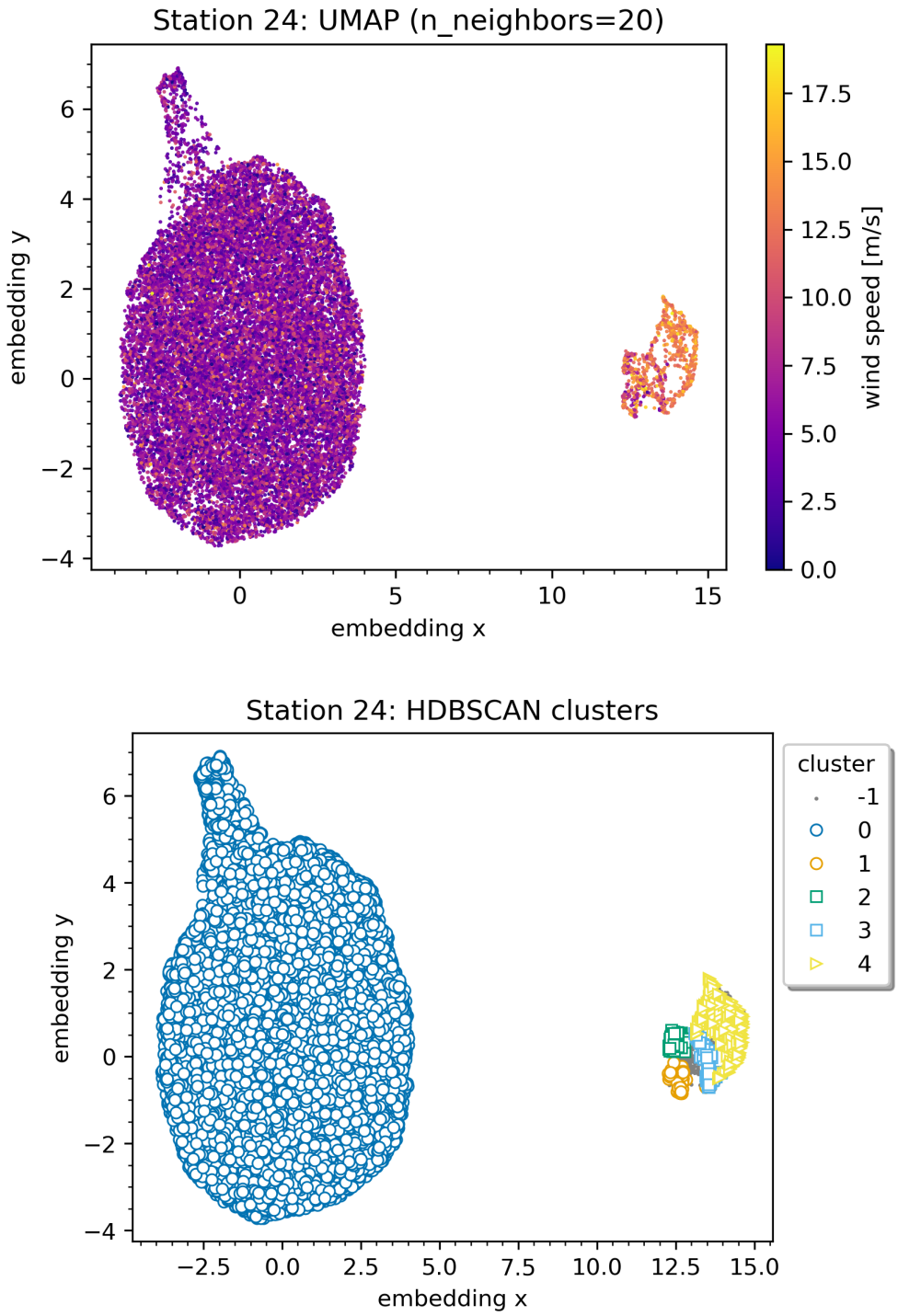


Figure A.74.: UMAP representation of the latent vectors of station 24 shallow- and force-triggered events with the corresponding wind speed encoded by color (top) and overlayed with clusters found by HDBSCAN (bottom). Cluster -1 denotes those events that are not assigned to any cluster.

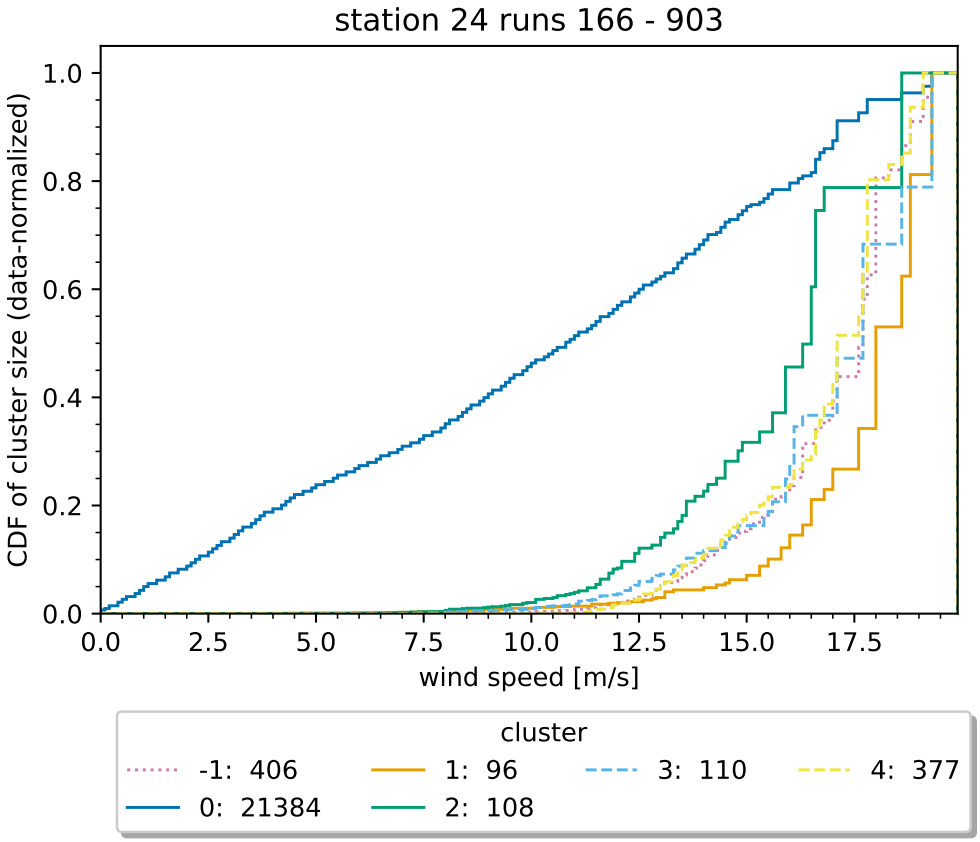


Figure A.75.: Station 24: CDF of the cluster size as a function of the wind speed (data-normalized). The total number of events per cluster is shown in the legend.

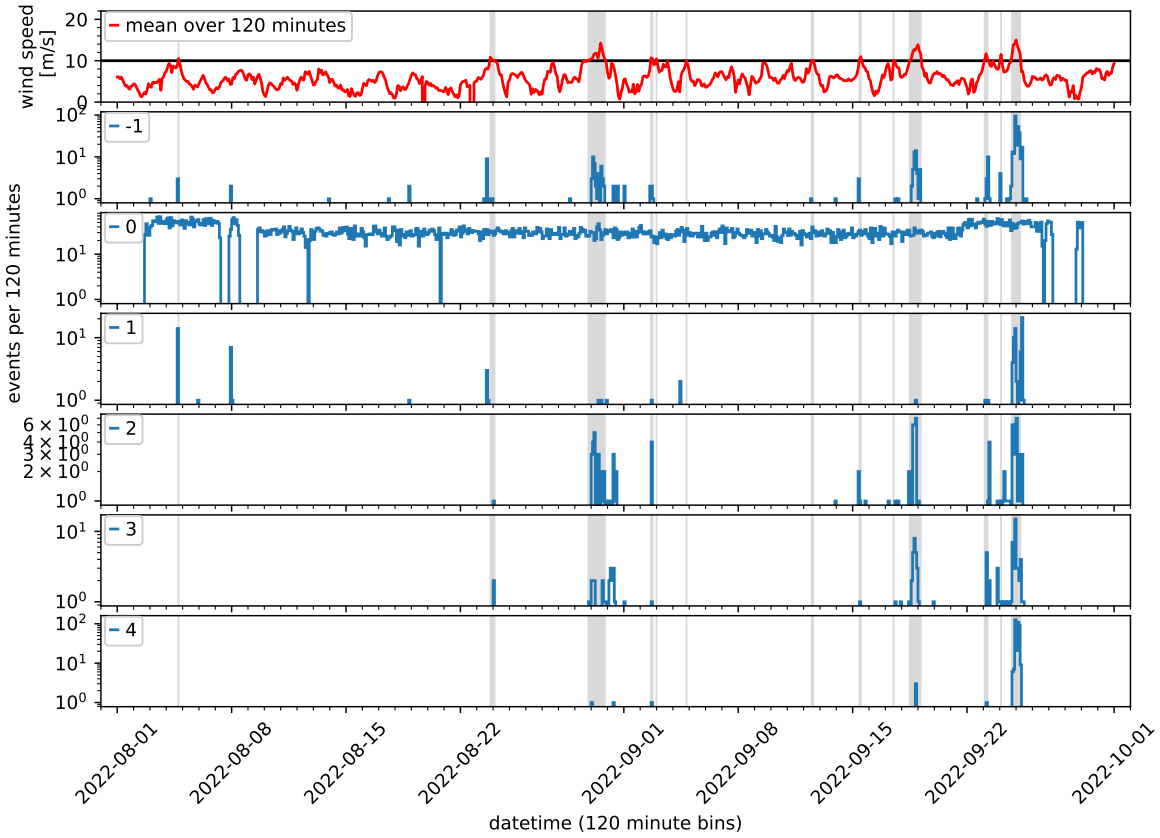
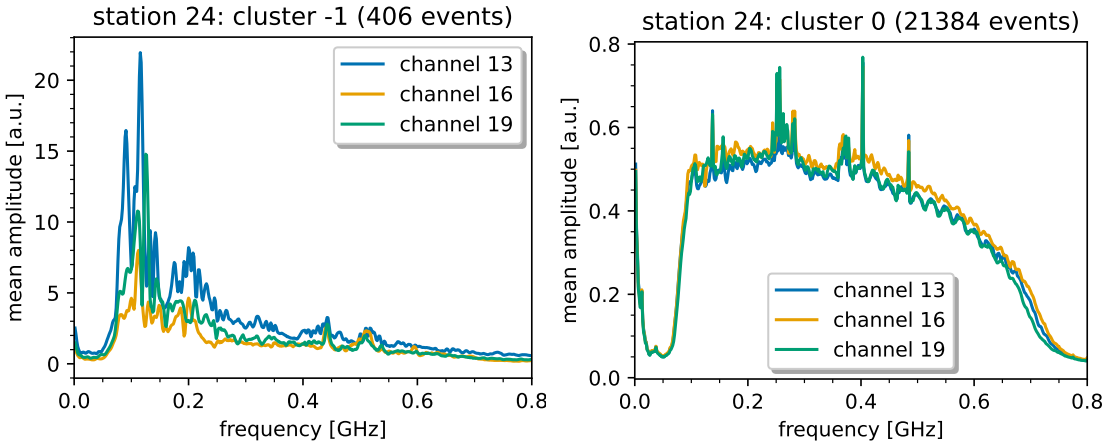


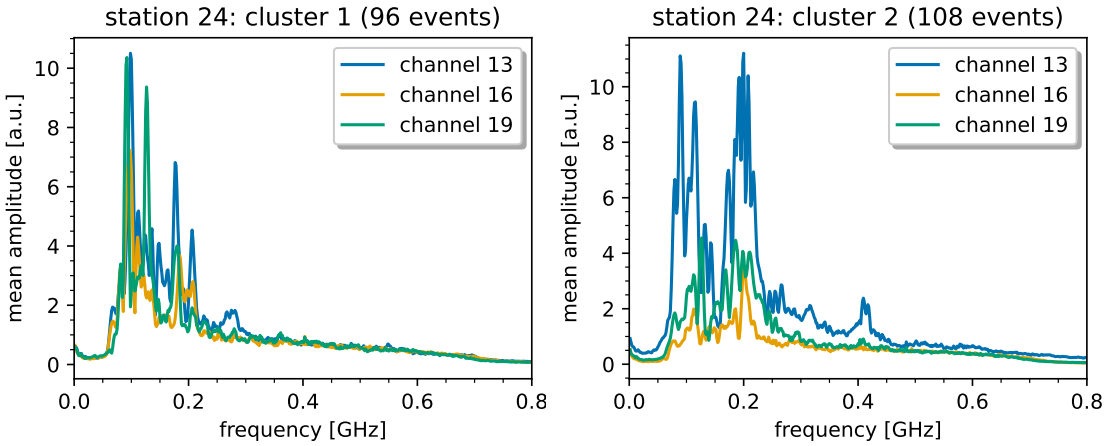
Figure A.76.: Station 24: event rate per cluster found by HDBSCAN as well as the wind speed. Time intervals with wind speed $\geq 10\,\mathrm{m/s}$ are shaded in gray. Time is given in UTC.

A.2.4.1. Mean frequency spectra per cluster



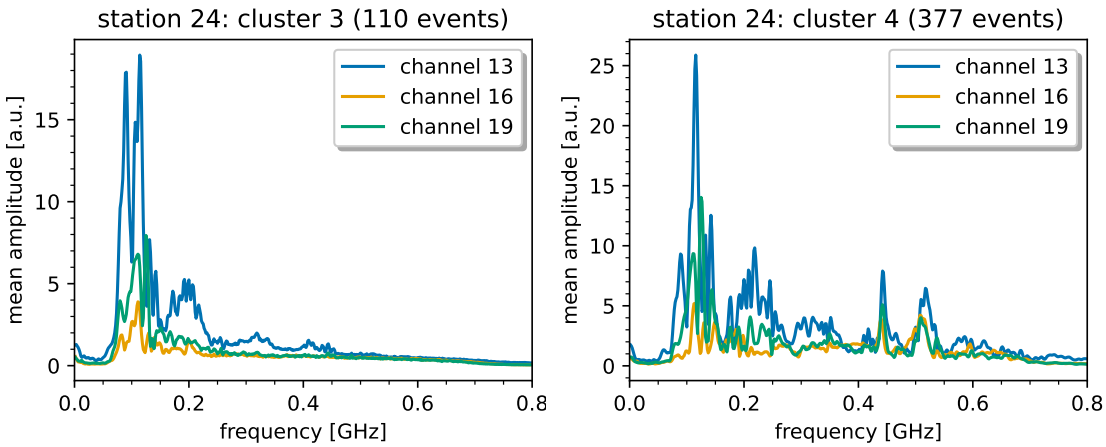
(a) **(b)**

Figure A.77.: Mean frequency spectra per channel for a) cluster −1 and b) cluster 0 (station 24).



(a) **(b)**

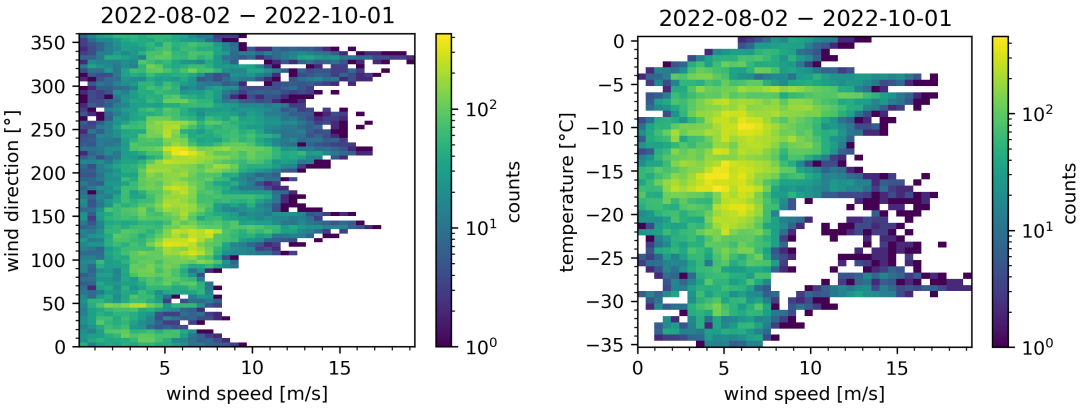
Figure A.78.: Mean frequency spectra per channel for a) cluster 1 and b) cluster 2 (station 24).



(a) (b)

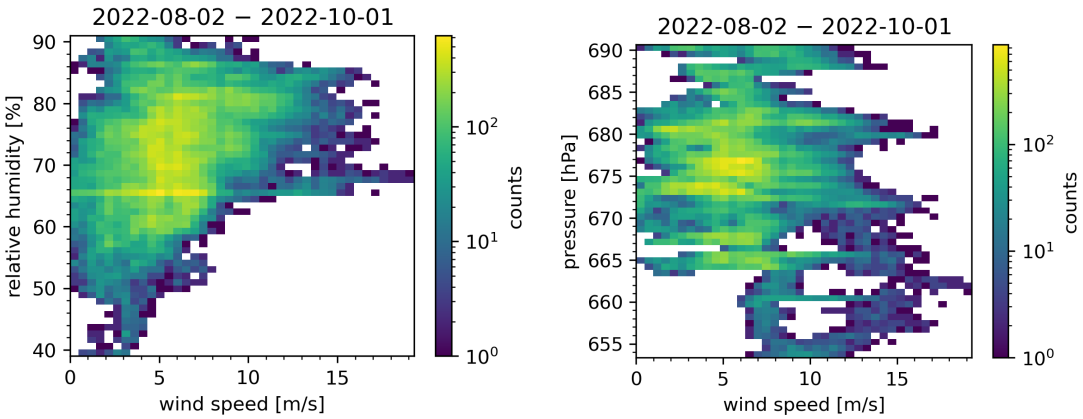
Figure A.79.: Mean frequency spectra per channel for a) cluster 3 and b) cluster 4 (station 24).

A.3. Various weather quantities vs. wind speed



(a) Wind direction vs. wind speed **(b)** Temperature vs. wind speed

Figure A.80.: 2D histograms of a) the wind direction and b) the temperature as a function of the wind speed. A wind direction of 0 ° represents true north.



(a) Relative humidity vs. wind speed **(b)** Barometric pressure vs. wind speed

Figure A.81.: 2D histograms of a) the relative humidity and b) the barometric pressure as a function of the wind speed.

A.4. Wind event discrimination

A.4.1. Model architecture

Layer	Name	Output shape	Input
InputLayer	`input_1`	(128, 128, 1)	-
GaussianNoise	`gaussian_noise`	(128, 128, 1)	`input_1`
Conv2D + LReLU	`conv_1_1`	(128, 128, 32)	`gaussian_noise`
Conv2D + LReLU	`conv_1_2`	(64, 64, 32)	`conv_1_1`
Dropout	`dropout_1`	(64, 64, 32)	`conv_1_2`
Conv2D + LReLU	`conv_2_1`	(64, 64, 64)	`dropout_1`
Conv2D + LReLU	`conv_2_2`	(32, 32, 64)	`conv_2_1`
Dropout	`dropout_2`	(32, 32, 64)	`conv_2_2`
Conv2D + LReLU	`conv_3_1`	(32, 32, 128)	`dropout_2`
Conv2D + LReLU	`conv_3_2`	(16, 16, 128)	`conv_3_1`
Dropout	`dropout_3`	(16, 16, 128)	`conv_3_2`
Conv2D + LReLU	`conv_4_1`	(16, 16, 256)	`dropout_3`
Conv2D + LReLU	`conv_4_2`	(8, 8, 256)	`conv_4_1`
Flatten	`flatten`	(16384)	`conv_4_1`
Dense + LReLU	`dense_1`	(64)	`flatten`
Dense + Linear	`z_mean`	(6)	`dense_1`
Dense + Linear	`z_log_var`	(6)	`dense_1`
Sampling	`z`	(6)	`z_mean` + `z_log_var`

Table A.6.: Structure of the encoder of the VAE used for wind event discrimination. The first column specifies the respective layer and its corresponding activation function if the layer has one. LReLU denotes the LeakyReLU activation. The sampling layer calculates the latent variable z using the outputs of the `z_mean` and `z_log_var` layers and a random number generator according to the reparameterization trick (see Equation 4.27). For a 2D convolutional layer, the first two dimensions of the output shape denote the spatial dimensions and the last dimension denotes the number of channels/filters.

Layer	Name	Output shape	Input
InputLayer	`input_2`	(6)	`z`
Dense + LReLU	`dense_2`	(16384)	`input_2`
Reshape	`reshape`	(8, 8, 256)	`dense_2`
Dropout	`dropout_4`	(8, 8, 256)	`reshape`
Conv2DTr + LReLU	`convtr_1_1`	(8, 8, 256)	`dropout_4`
Conv2DTr + LReLU	`convtr_1_2`	(16, 16, 256)	`convtr_1_1`
Dropout	`dropout_5`	(16, 16, 256)	`convtr_1_2`
Conv2DTr + LReLU	`convtr_2_1`	(16, 16, 128)	`dropout_5`
Conv2DTr + LReLU	`convtr_2_2`	(32, 32, 128)	`convtr_2_1`
Dropout	`dropout_6`	(32, 32, 128)	`convtr_2_2`
Conv2DTr + LReLU	`convtr_3_1`	(32, 32, 64)	`dropout_6`
Conv2DTr + LReLU	`convtr_3_2`	(64, 64, 64)	`convtr_3_1`
Dropout	`dropout_7`	(64, 64, 64)	`convtr_3_2`
Conv2DTr + LReLU	`convtr_4_1`	(64, 64, 32)	`dropout_7`
Conv2DTr + LReLU	`convtr_4_2`	(128, 128, 32)	`convtr_4_1`
Conv2DTr + max(ReLU, 1)	`output`	(128, 128, 1)	`convtr_4_2`

Table A.7.: Structure of the decoder of the VAE used for wind event discrimination. The first column specifies the respective layer and its corresponding activation function if the layer has one. LReLU denotes the LeakyReLU activation and Conv2DTr means 2D transposed convolutional layer. The activation function of the output layer is a ReLU limited to 1. For a 2D convolutional layer, the first two dimensions of the output shape denote the spatial dimensions and the last dimension denotes the number of channels/filters.

A.4.2. Example events with low and high loss

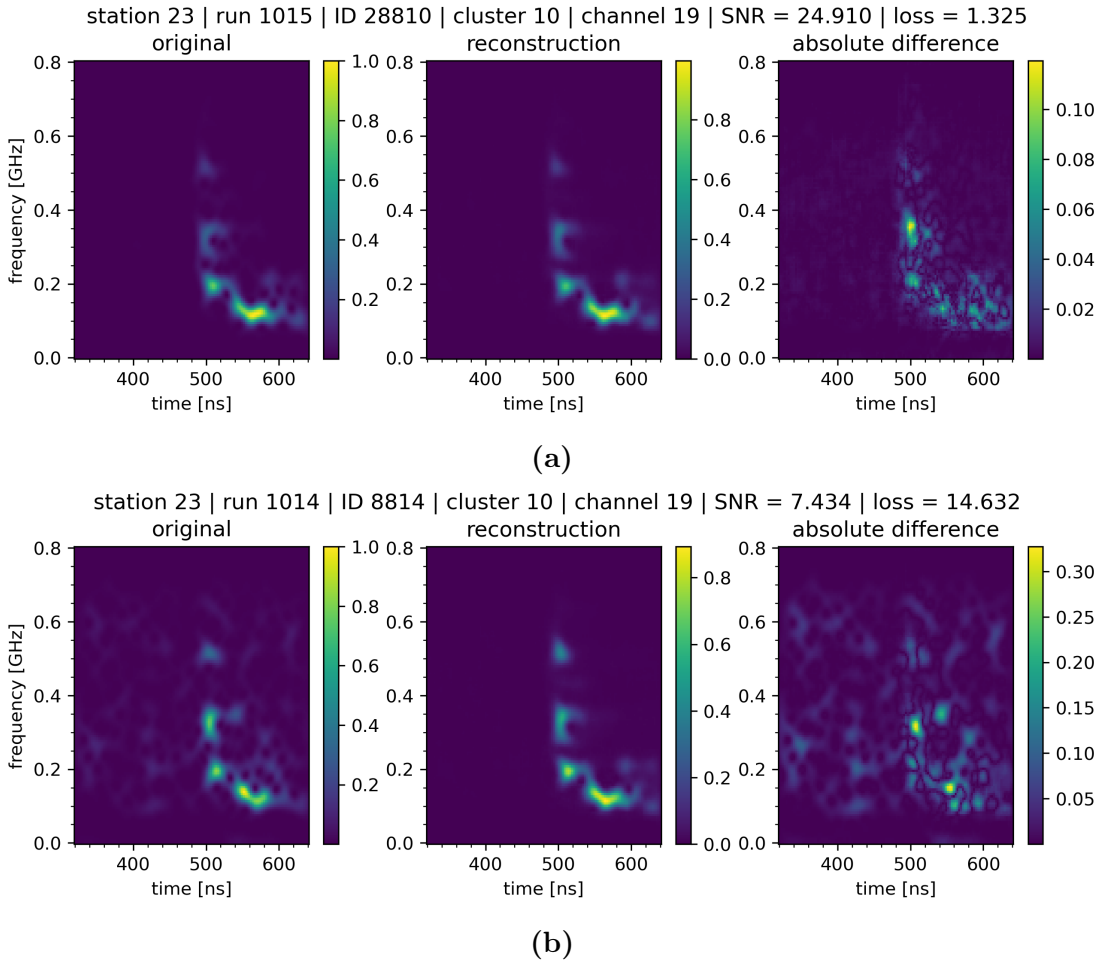
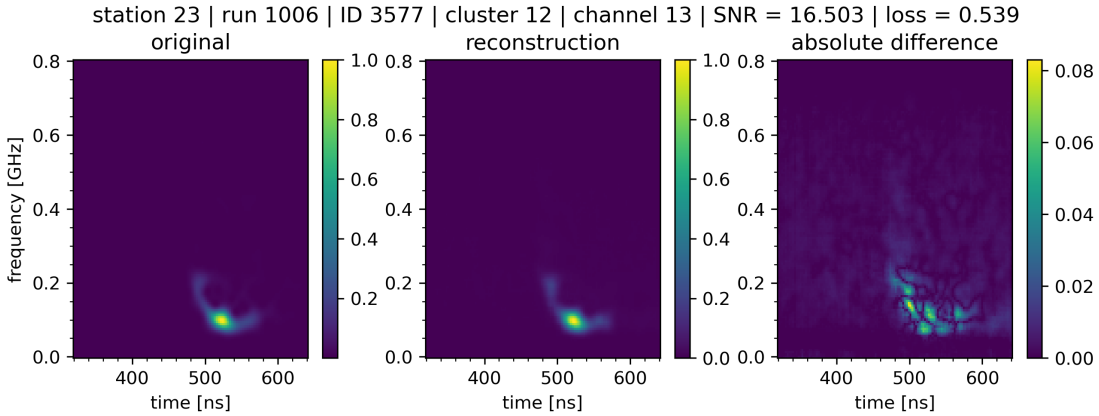
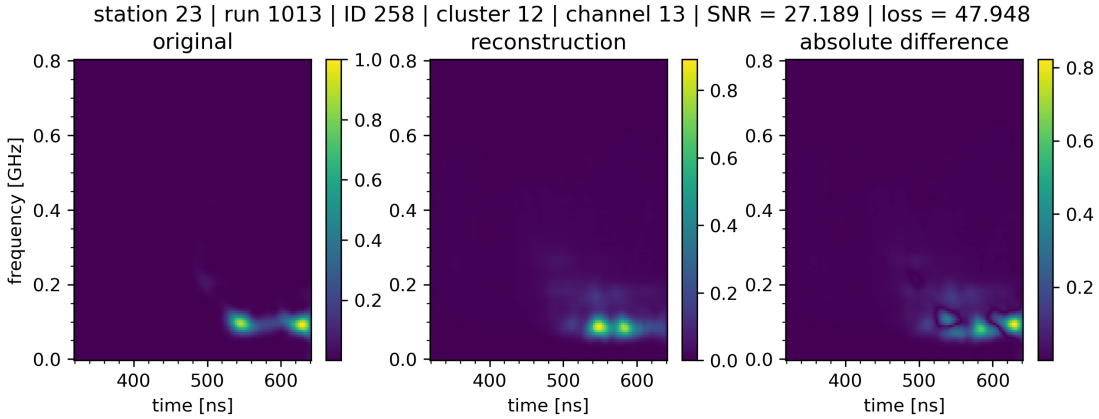


Figure A.82.: Example events of cluster 10 of station 23 with a) low and b) high loss. From left to right: spectrogram of the waveform (prepared as input for the variational autoencoder (VAE)), output reconstruction of the VAE, absolute of the difference between input and output.

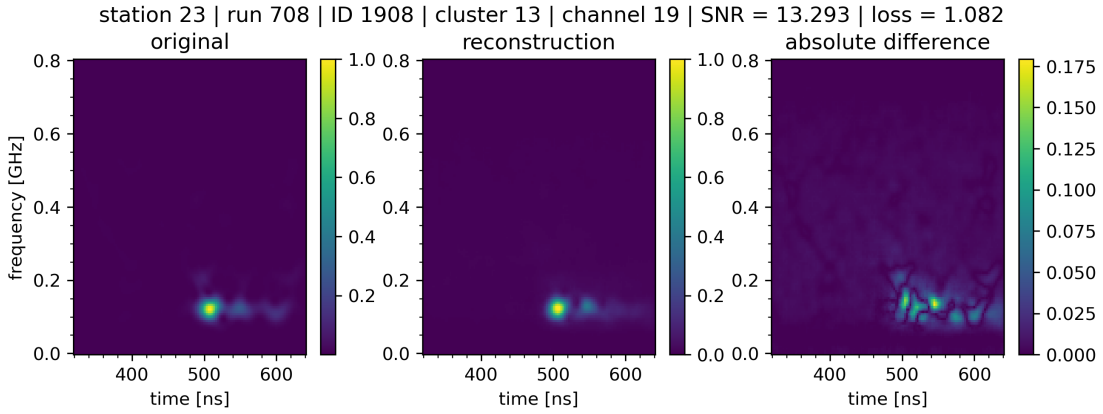


(a)

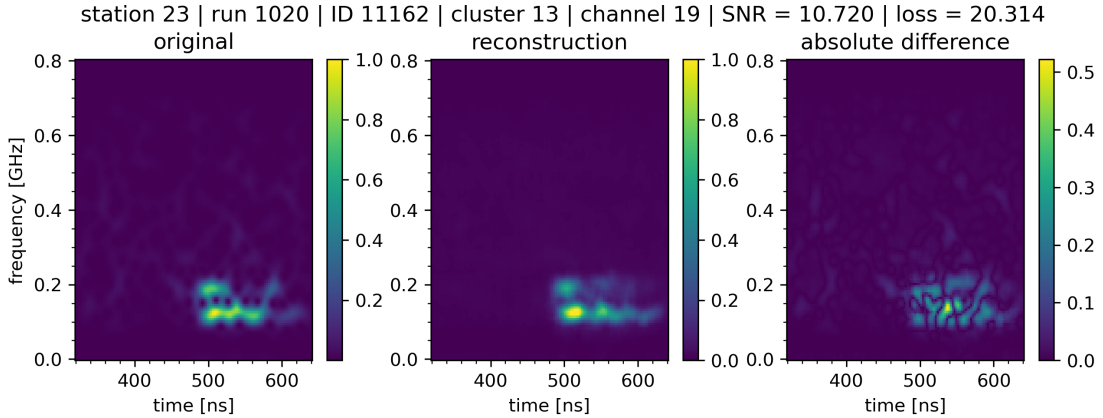


(b)

Figure A.83.: Example events of cluster 12 of station 23 with a) low and b) high loss. From left to right: spectrogram of the waveform (prepared as input for the variational autoencoder (VAE)), output reconstruction of the VAE, absolute of the difference between input and output.

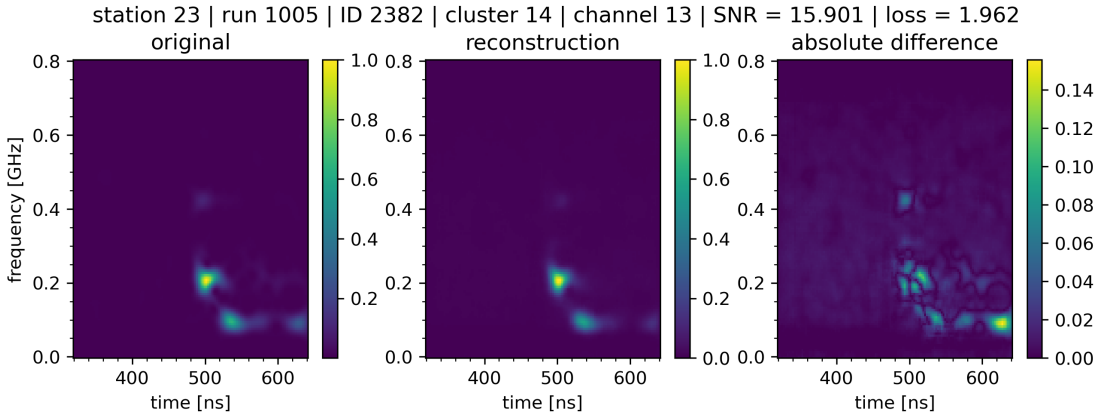


(a)

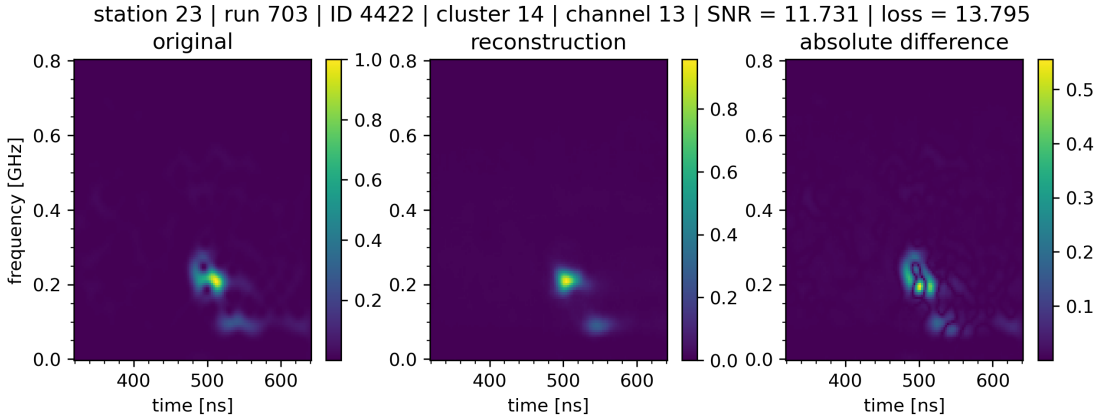


(b)

Figure A.84.: Example events of cluster 13 of station 23 with a) low and b) high loss. From left to right: spectrogram of the waveform (prepared as input for the variational autoencoder (VAE)), output reconstruction of the VAE, absolute of the difference between input and output.



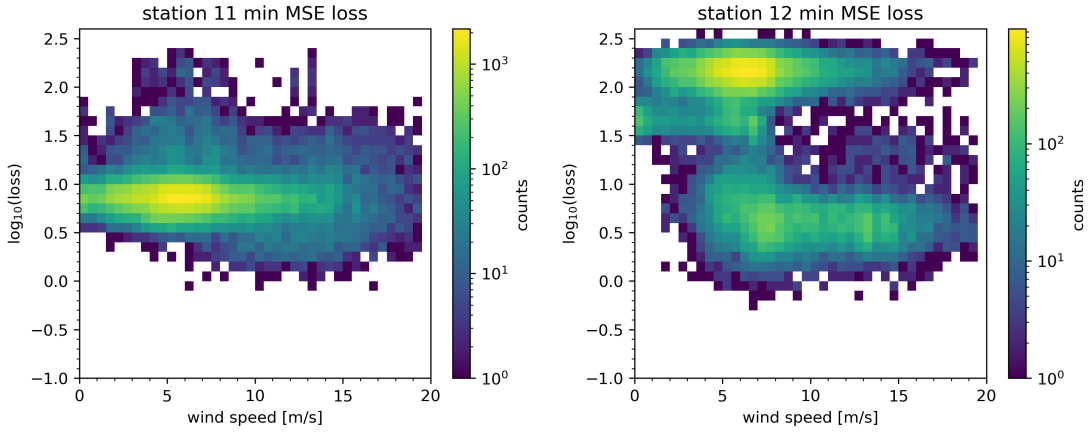
(a)



(b)

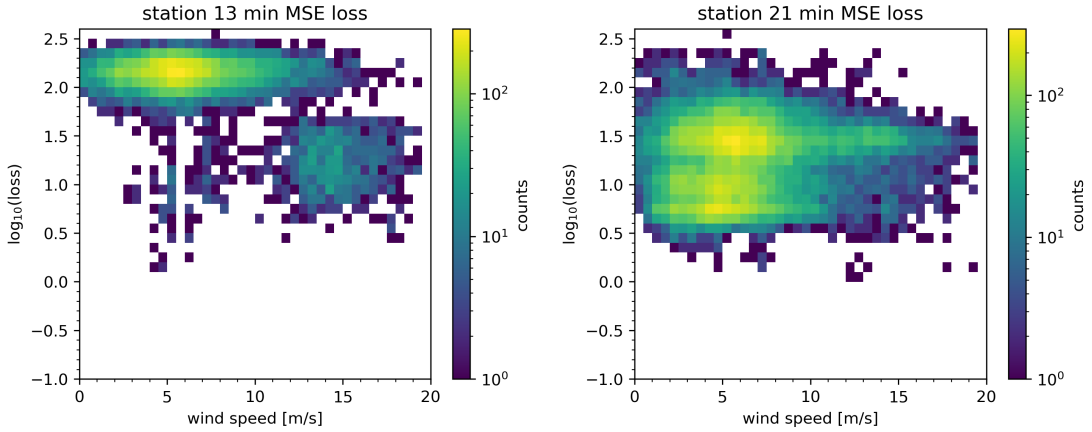
Figure A.85.: Example events of cluster 14 of station 23 with a) low and b) high loss. From left to right: spectrogram of the waveform (prepared as input for the variational autoencoder (VAE)), output reconstruction of the VAE, absolute of the difference between input and output.

A.4.3. Loss as a function of wind speed per station



(a) (b)

Figure A.86.: 2D histograms of $\log_{10}$(loss) vs. wind speed of shallow-triggered events for a) station 11 and b) station 12. The stations are evaluated on the discriminator that is trained exclusively on wind events of station 23. The loss is calculated by taking the minimum over all three channels.



(a) (b)

Figure A.87.: 2D histograms of $\log_{10}$(loss) vs. wind speed of shallow-triggered events for a) station 13 and b) station 21. The stations are evaluated on the discriminator that is trained exclusively on wind events of station 23. The loss is calculated by taking the minimum over all three channels.

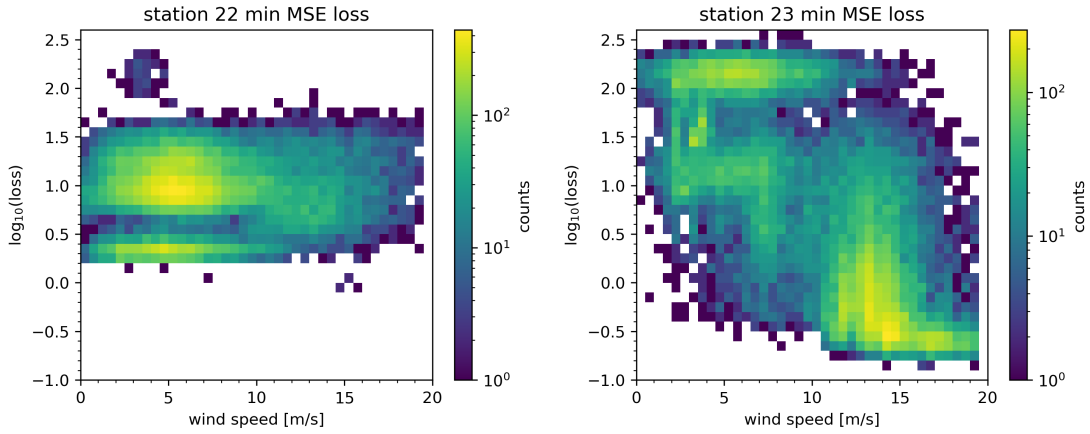


Figure A.88.: 2D histograms of $\log_{10}$(loss) vs. wind speed of shallow-triggered events for a) station 22 and b) station 23. The stations are evaluated on the discriminator that is trained exclusively on wind events of station 23. The loss is calculated by taking the minimum over all three channels.

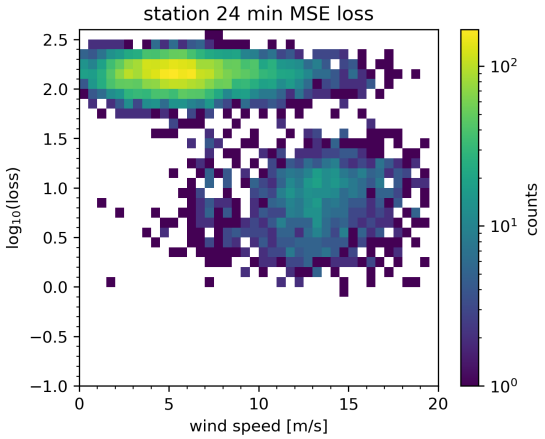


Figure A.89.: 2D histogram of $\log_{10}$(loss) vs. wind speed of shallow-triggered events for station 24. The station is evaluated on the discriminator that is trained exclusively on wind events of station 23. The loss is calculated by taking the minimum over all three channels.

Acknowledgements

I would like to thank Anna and Jakob for supervising me and for always giving great advice. Furthermore, I want to thank the DESY-ECAP RNO-G group for explaining various topics and for making the group meeting an interesting and fun time. Many thanks to all the people, who provided valuable comments and feedback for this thesis.

The time at ECAP was very enjoyable and therefore I would like to thank all the people at ECAP for interesting discussions, presentations, coffee breaks, BBQs, the Feuerzangenbowle, memes, and especially for fun and definitely pro-level table tennis matches (still looking forward to the Grand ECAP Table Tennis Tournament).

Finally, I want to thank my family and friends for always supporting me at all times.

Erklärung

Hiermit erkläre ich, dass ich die hier vorliegende Arbeit selbst verfasst habe. Es wurden nur die in dieser Arbeit angegebenen Quellen und Hilfsmittel verwendet.

Erlangen, 04.12.2023 Philipp Laub