Towards the Automatic Analysis of Ceilometer Backscattering Profiles using Unsupervised Learning

Michael Dammann
Hamburg University of Applied Sciences
michael.dammann@haw-hamburg.de

Ina Mattis
Deutscher Wetterdienst
ina.mattis@dwd.de

Michael Neitzke
Hamburg University of Applied Sciences
michael.neitzke@haw-hamburg.de

Ralf Möller
University of Lübeck
moeller@uni-luebeck.de

Abstract

Ceilometers use a laser beam to capture certain phenomena in the atmosphere like clouds, precipitation, or aerosol layers. These measurements can be visualized in so-called *quick looks* that at the moment are mostly analyzed manually by meteorology experts. In this work, we illustrate the path towards the automatic analysis of quick looks by using a hybrid approach combining an image segmentation algorithm with unsupervised representation learning and clustering. We present a first proof of concept and give an outlook on possible future work.

1 Introduction

The German Weather Service DWD (*Deutscher Wetterdienst*) operates a large ceilometer network with a size of more than 180 sites [1]. A ceilometer is a stationary LIDAR system for the measurement of atmospheric phenomena like clouds, aerosols, and precipitation. Its basic principle is measuring the backscattered light of a vertically pointing laser beam with a wavelength of 1064 nm. Depending on the matter occurring in the atmosphere, the amount of backscattered light varies. It is higher for rain clouds than for aerosols, for example. The altitude where backscattering occurred can be derived by using the travel time of the emitted and then backscattered photons.

Based on these backscattering profiles, so-called quick looks are generated, which essentially are the application of a defined color map to the measured values, usually spanning a single day. An example is given in Fig. 1; the y-axis refers to the altitude and the x-axis to the time of the day. For this example, the occurring phenomena are annotated, namely *clouds*, *saharan dust*, and *background aerosol*. At the moment, the ceilometer data is manually analyzed by meteorology experts. Given the large stream of data from all the different locations, it is not possible for the limited amount of experts (and time) to examine every single backscattering profile. For this reason, in this work, a path towards the automatic analysis of ceilometer backscattering profiles using deep learning is proposed. The possible pathway to climate impact of such a system is twofold:

- Climate science insights by applying deep learning, like frequent, but not necessarily obvious, phenomena in certain locations or at specific altitudes, which cannot be derived manually because of the large amount of data.
- Automatic detection of situations with specific meteorological conditions for more detailed case studies, e.g., on aerosol-cloud interactions.

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2022.

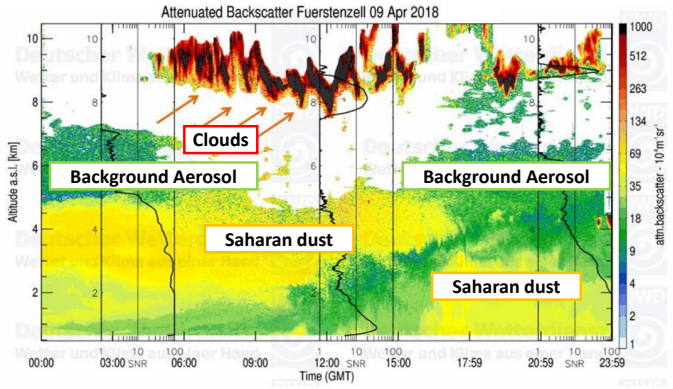


Figure 1: A ceilometer quick look from the location of Fürstenzell, April 9, 2018. Adapted from [2].

2 Hybrid Ceilometer Measurement Categorization Approach - A Prototype

The ceilometer data and resulting quick looks are unlabeled, i.e., annotations tagging the individual phenomena like clouds do not exist. Thus, we have to make use of algorithms that are suitable for unlabeled data.

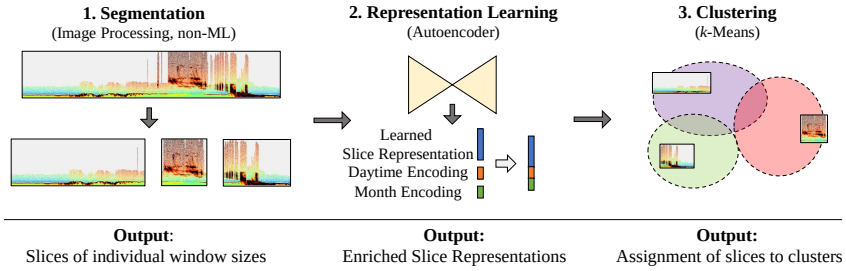


Figure 2: High-level overview of our hybrid clustering approach.

Our approach for a first prototype is visualized in Fig. 2. For our idea, we follow the intuition of a meteorology expert. First, the expert takes a look at the quick look, differentiates the occurring phenomena based on shape, position, and backscattering values (colors) and then groups them into different phenomenon classes. We use a segmentation algorithm from "classic" image processing (i.e., no machine learning) like Felzenszwalb-Huttenlocher [3] (FH) or Simple Linear Iterative Clustering [4] (SLIC) to imitate this behavior to automatically detect the phenomena occurring in a quick look. For example, clouds have a distinct dark color and shape, which makes them detectable relatively easily by an image segmentation algorithm. We use the resulting segment boundaries to generate *slices* of the detected phenomena as visualized in the bottom left of Fig. 2. In step 2, vector representations (colored blue in Fig. 2) of the slices are learned using an autoencoder [5]. Since the daytime information is lost during the slicing step, we further incorporate it into the representation by concatenating a daytime encoding (orange) to the autoencoder representation. The month information (green) is also added to the representation to possibly capture seasonal patterns better. Finally, the resulting vector representations are clustered using k-means [6, 7]. More details on the implementation of the segmentation, autoencoder, and clustering steps and more insights into the results can be found in the Appendix under A, B and C, respectively.

We use $k = 8$ clusters for our prototype and visualize the results in 2-d using PCA [8], see Fig. 3. In general, it is apparent how reasonable clusters for the slice representations are found. For example, the top left part of the projection captures low-altitude phenomena like fog (cluster colored in blue), precipitation can be found in the cluster colored in purple, and clear skies or skies with some clouds in the grey and brown clusters. During noon time, phenomena of high backscattering values at high altitudes can be observed, visible by the apparent dark colors in the right to top right region. The resulting clusters can be categorized by meteorology experts, possibly circumventing the labeling process of the whole ceilometer dataset.

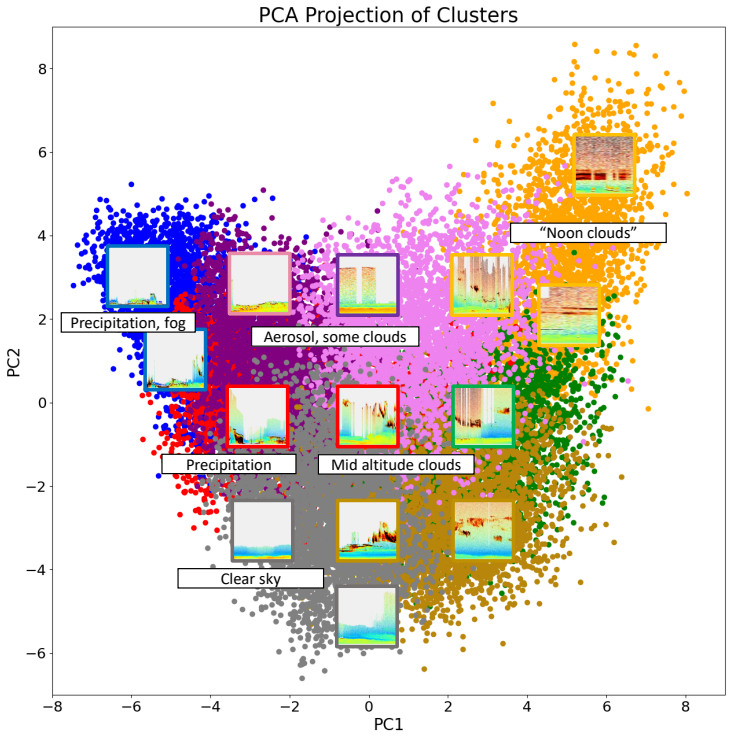


Figure 3: Example segments for several coordinates in the PC1 and PC2 projection space and interpretation thereof.

3 Conclusion and Outlook

We proposed an approach for the unsupervised, automatic categorization of ceilometer quick looks. In a first prototype, we showed that the approach leads to a reasonable clustering of meteorological phenomena.

While other works applying machine learning to ceilometer data exist, a "single-task" trend can be observed. These works consist of using supervised learning to detect precipitation and fog [9], combining ceilometer and sky camera data to train a classifier on cloud-type detection [10] and applying machine learning to determine planetary boundary layer heights [11, 12]. On the contrary, we propose a more "holistic", vision-based approach with the aim of detecting all possibly occurring phenomena.

Our approach described in Sec. 2 can also be considered a general framework for future work. I.e., while the single steps stay the same, each could be filled with a different algorithm or method. For example, regarding the second step, self-supervised methods [13, 14] for more robust vector representations could be applied. As for the clustering step, there exist deep learning clustering approaches that combine representation learning and clustering in a single method [15, 16], which would also be suitable for the ceilometer use case described in this thesis.

Overall, we consider our prototype the be a successful proof of concept for our unsupervised hybrid ceilometer quick look categorization framework. The next steps will consist of adding more advanced deep learning methods to the approach. We also plan to acquire a test dataset consisting of labeled ceilometer quick looks to also - in addition to the qualitative evaluation above - quantitatively evaluate the approach. Technically, the test dataset will consist of segmentation masks, making it possible to evaluate the results both in regard to how well the single phenomena are segmented (i.e., the detected *shape*) as well as to how successful their cluster assignment (*class*) is. Using the test dataset, mean Average Precision (mAP) will be applied to evaluate the approach. It presents a metric that captures the quality of both segmentation and clustering results which is crucial to successfully detect, e.g., aerosol-cloud interactions which could then be studied in a further step.

Acknowledgments and Disclosure of Funding

The work on which this publication is based on was funded by the German Federal Ministry for Digital and Transport (Bundesministerium für Digitales und Verkehr, BMDV) under FE no. 50.0387/2020. The sole responsibility for the content lies with the authors.

References

[1] DWD - Ceilometer data and trajectories. `https://www.dwd.de/EN/research/projects/ceilomap/ceilomap_node.html`. Accessed: 2022-09-17.

[2] Werner Thomas, Harald Flentje, Ina Mattis, and Gerhard Müller. How ceilometers detect saharan dust events. `https://www.dwd.de/DWD/forschung/projekte/ceilomap/files/Saharan_dust_example_en.pdf`, May 2018. Accessed: 2022-09-15.

[3] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vis.*, 59(2):167–181, 2004.

[4] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[5] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[6] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[7] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[8] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[9] Yong-Hyuk Kim, Seung-Hyun Moon, and Yourim Yoon. Detection of precipitation and fog using machine learning on backscatter data from lidar ceilometer. *Applied Sciences*, 10(18):6452, Sep 2020.

[10] J. Huertas-Tato, F. J. Rodríguez-Benítez, C. Arbizu-Barrena, R. Aler-Mur, I. Galvan-Leon, and D. Pozo-Vázquez. Automatic cloud-type classification based on the combined use of a sky camera and a ceilometer. *Journal of Geophysical Research: Atmospheres*, 122(20):11,045–11,061, 2017.

[11] Jennifer Sleeman, Milton Halem, Zhifeng Yang, Vanessa Caicedo, Belay Demoz, and Ruben Delgado. A deep machine learning approach for lidar based boundary layer height detection. In *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 3676–3679, 2020.

[12] Jin Ye, Lei Liu, Qi Wang, Shuai Hu, and Shulei Li. A novel machine learning algorithm for planetary boundary layer height estimation using aeri measurement data. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 1597–1607. PMLR, 2020.

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020.

[15] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[16] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5147–5156, 2016.

A Segmentation

As described, in the first step the backscattering profile is segmented using "classical" image processing algorithms. These segments are then used to create individual window sizes, depending on the size of the respective phenomenon.

An open question is the choice of input representation. Naturally, the measured backscattering values could be used as input. Alternatively, the RGB values resulting from applying a colormap to the backscattering value could also be utilized. This could perhaps be closer to human intuition since human experts rely on these colors to make an accurate classification of the displayed backscattering profile. A comparison of these two options is given in Sec. A.1.

When it comes to segmentation algorithms, two widespread approaches are Felzenszwalb-Huttenlocher [3] (FH) and Simple Linear Iterative Clustering [4] (SLIC). FH is a graph-based algorithm, with the pixels being the vertices and the edge weights being a non-negative dissimilarity measure (based on color and position). Iteratively, components (sub-graphs) are constructed that have a small internal difference (e.g., neighbored parts of the image in similar color) and a high difference to other constructed components. On the other hand, SLIC is an application of the k-means algorithm, using the color (or grayscale) values and the position of the pixels for clustering. Both are evaluated for the task of backscattering profile segmentation in Sec. A.2.

The resulting dataset that will be used for training an autoencoder in step 2 of the approach is discussed in Sec. A.3.

A.1 Colors vs. Backscattering

Looking at the color scale on the right in Fig. 1, it can be seen that the applied color map is logarithmic. This means that the backscattering values in the red and black regions of the image are roughly ten times larger than in the yellow region and a hundred times larger than in the green regions, respectively. Using these values as-is will overemphasize the red and black parts dramatically while all other regions more or less get lost as noise. This would distort both the segmentation algorithm and the autoencoder results as they would overvalue the red and dark parts. This should not be the case, since all other regions of a profile are equally important. An example is given in Fig. 4, where the "raw" values are used for segmentation with FH.

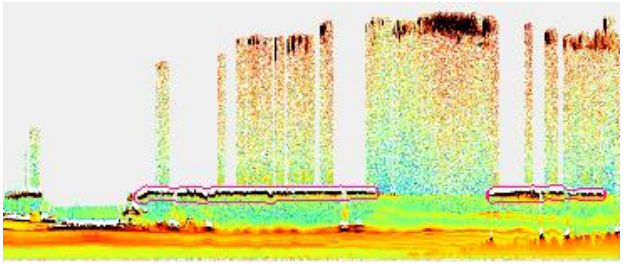
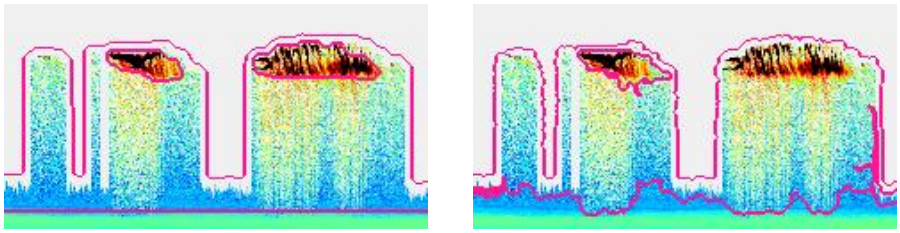


Figure 4: Using "raw" backscattering measurements overvalues high backscattering phenomena. (Leipzig, September 22, 2017)

Here, all segments detected by FH (circled in magenta color) are those with very high backscattering values. Some high backscattering phenomena like the clouds at higher altitudes are not even detected at all. The aerosol at low altitude (orange regions) is not detected either. The results for SLIC are comparable. For this reason, logarithmized values could be an appropriate preprocessing step. The other way, as mentioned, is to directly use the color values from the backscattering profile image. In a sense, colorizing can be seen as a conversion of backscattering values to (soft) multiclass affiliations like *80 % red*, *50 % green*, and *20 % blue*.

Fig. 5 shows a segmentation comparison between logarithmized backscattering values and colors using FH. It is apparent that using colors leads to better segmentation. The clouds (red/black, top), planetary boundary layer (green, bottom), and the background are separated well. By comparison, using the logarithmized backscattering results in much noisier segment boundaries. The cloud in



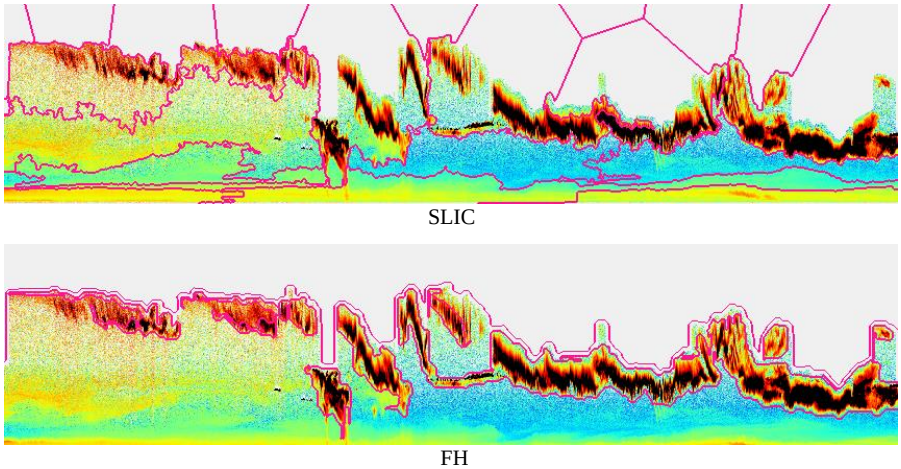
Colors Logarithmized Backscattering

Figure 5: Comparison between using colors and logarithmized backscattering values for segmentation with FH. (Hersfeld, August 6, 2018)

the top-right is not even detected at all. It should be noted that the results of FH and SLIC are very sensitive to their respective parameter settings. However, evaluating a wide range of parameters on a set of 30 backscattering profiles, it was determined that using colors essentially always led to better segmentation. For this reason, in the course of this work, color values will be used.

A.2 FH vs. SLIC

Fig. 6 shows a comparison between FH and SLIC segmentation for a given backscattering profile, using color values as discussed in Sec. A.1.



SLIC

FH

Figure 6: Comparison between FH and SLIC. (Leipzig, January 9, 2018)

It can be seen that - in general - FH works better. The cutouts are more precise and very accurate segments are generated. Using SLIC, the number of segments is parameterized (since it is based upon k-means). In general, this is a problem since the number of phenomena can vary (highly) from profile to profile. It also perhaps leads to the observed effect of all segments having a very similar area size, which is also a problem considering that the phenomena differ significantly in terms of size. SLIC does seem to work better in the green and yellow areas, however, there seems to be a degree of over-segmentation. Judging from these observations, FH will be used in this work to generate an individual window size for each phenomenon - keeping the option of further optimization for the detection of the "soft transitions" (like in Fig. 6 between green, yellow and blue) in mind. It would also be possible to only use the cut-out segment. However, using the whole height range is thought of providing vital context to the subsequent steps. More details on FH and the final parameters used can be found in Appendix A.4.

A.3 Resulting Dataset

Finally, the FH segments can be used to create individual window sizes. To achieve this, simply the minimal and maximal x value for a segment is determined and used to cut out the corresponding window from the profile. An example is given in Fig. 7.

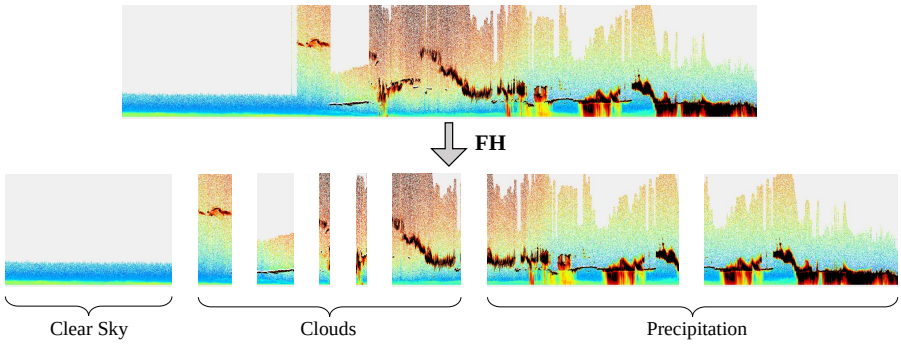


Figure 7: Example of individual window sizes for a single profile. (Leipzig, January 6, 2014)

It can be seen that the different phenomena are in general successfully separated. In this case, the three different types *clear sky*, *clouds*, and *precipitation* can be made out. Applying another set of parameters to FH, it is possible to segment (for example) the precipitation windows on the right into further smaller ones. However, this could also lead to over-segmentation in other places. Having evaluated them on 30 backscattering profiles, the current FH settings are considered to provide an overall good balance between over- and under-segmentation. Overlaps between segments are accepted since some phenomena can actually overlap (for example two stacked clouds) and many phenomena are fuzzy and transition smoothly between each other.

The DWD ceilometer data from Leipzig, ranging from the years 2014 to 2018, is used to create a dataset consisting of such individually sized windows. Fig. 8 shows a histogram presenting the distribution of slice durations in hours. Most slices are rather short in duration with the highest amount being in the 0 to 1 hour range. This is plausible since there often exists some overlap between segments and usually the many, smaller components of larger segments are detected separately as well. There exists a spike in the 23-24 hour range. This can be traced back to the FH algorithm which in some cases detects the whole day profile outline as a single segment. For now, these samples are left in the dataset, since in some cases it might be sensible to use the whole day as a single phenomenon. For example, when the whole day is constantly cloudy it is perfectly fine to detect all 24 hours as a single coherent segment.

Table 1 displays some statistical properties of the slice durations. The median being lower than the mean again reflects there being outliers to the right ("whole day segments"). 25 % of the slices are at maximum ca. 50 minutes long (0.82 hours), again showing that most slices are rather short. The wide distribution of window sizes (i.e., not all window sizes being the same) support the hypothesis that individual window sizes are the right choice for the detection of meteorological phenomena.

Table 1: Descriptive Statistics on slice durations/lengths.

Nmb. of Slices	26,034
Mean (h)	6.23
Std. deviation (h)	7.57
Median (h)	3.01
25 % percentile (h)	0.82
75 % percentile (h)	7.34

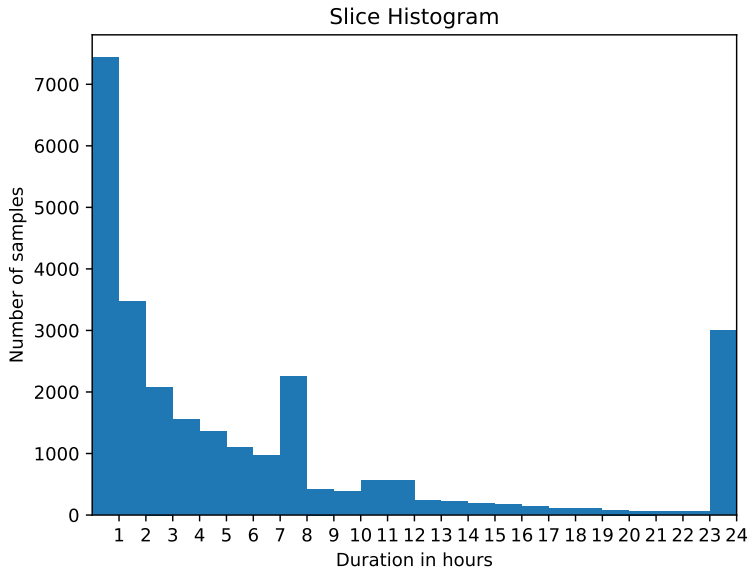


Figure 8: Histogram of slice durations.

A.4 Implementation

The `scikit-image` 0.19.1 implementation of FH was used:
`skimage.segmentation.felzenszwalb`.
The following parameters were applied: scale=1500, sigma=2, min_size=200.

B Representation Learning

In the next step, low-dimensional representations of the slices are learned using a convolutional autoencoder [5], the architecture is visualized in Fig. 9. Since (these kinds of) neural networks require a fixed-size input, all slices are resized to 128x128 pixels. The slice dataset (see Sec. A.3) was split 80%-10%-10% into training, validation, and test datasets. Several latent space sizes were evaluated and the dimensionality of 128 provided the best results. Lower dimensionalities led to poorer reconstructions and higher dimensionalities did not provide better results.

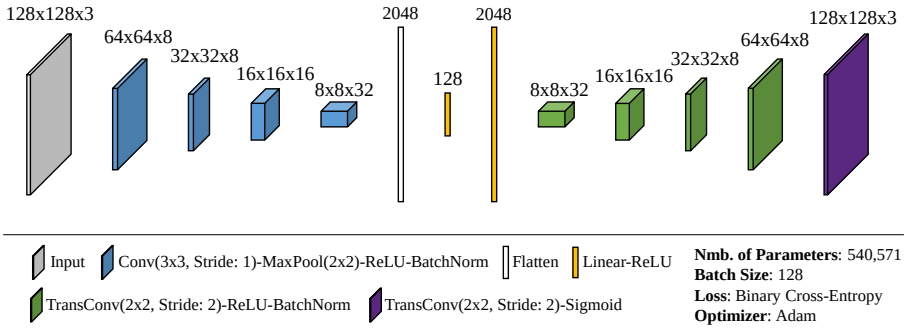


Figure 9: Architecture of the used autoencoder.

Some reconstructions are shown in Fig. 10. The reconstruction capture all the important parts of the input: shapes, colors, and positions. The reconstruction is lossy, some of the details are not reconstructed. This is expected and even deliberate since autoencoders belong to the class of

lossy compression methods. Losing some of the details prevents overfitting and leads to better generalization.

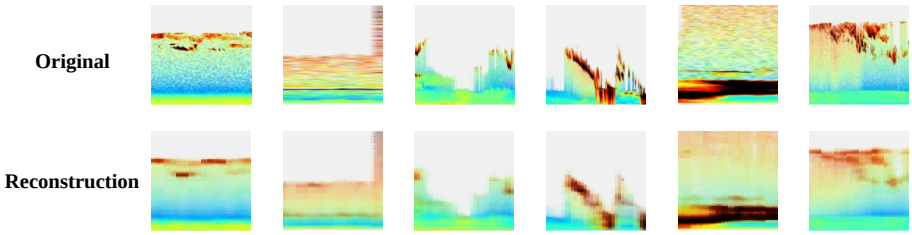


Figure 10: Autoencoder reconstruction examples.

When cutting out the segments from the whole profile and then resizing them to 128x128 pixels, information about when the phenomenon takes place and for how long is lost. For this reason, the representation is further augmented by concatenating this information to the autoencoded vector. The procedure is depicted in Fig. 11.

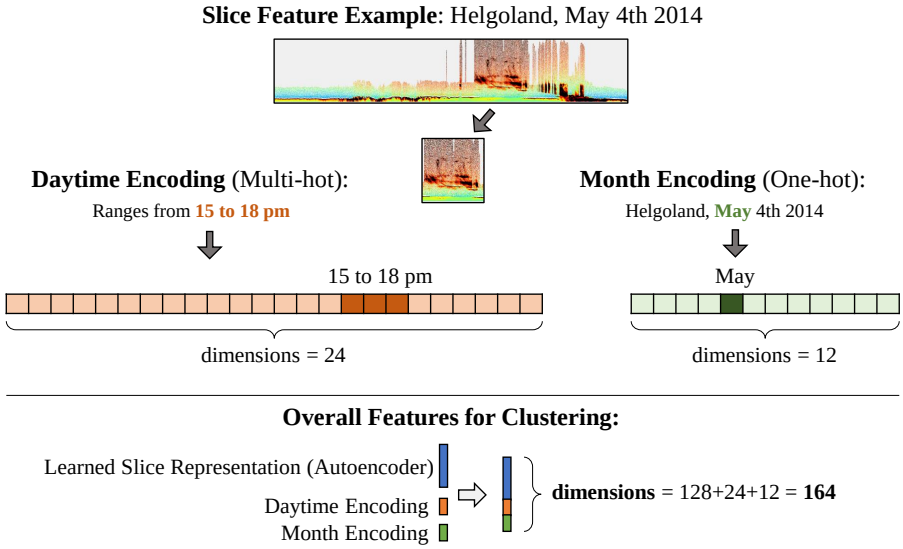


Figure 11: Example of feature construction for clustering for a single slice (i.e., single data point).

A 24-dimensional vector represents the hours of the day the detected segment took place in. For example, if the segment spans from 3 to 6 pm, the indices from 15 to 17 are set to 1, and all others are set to 0 (*multi-hot encoding*, orange). The times are rounded, so if the segment would have ended at 6:45 pm, index 18 would also be set to 1. The month is also encoded, using a 12-dimensional *one-hot encoded* vector (green). The concatenation of all three vectors forms the representation used for clustering in the next step.

B.1 Implementation

`pytorch` 1.10.1+cu113 and `pytorch-lightning` 1.5.7 were used for the neural network implementation. The experiments were carried out on an NVIDIA GeForce 3060 Laptop GPU.

C Clustering

In the final step, the obtained representations are clustered using k-means [7, 6]. Overall, the three basic phenomena *cloud*, *aerosol*, and *precipitation* can occur in a segment. Thus, using the cardinality of the powerset of these three phenomena, $k = 2^3 = 8$ clusters are chosen.

C.1 Cluster visualization

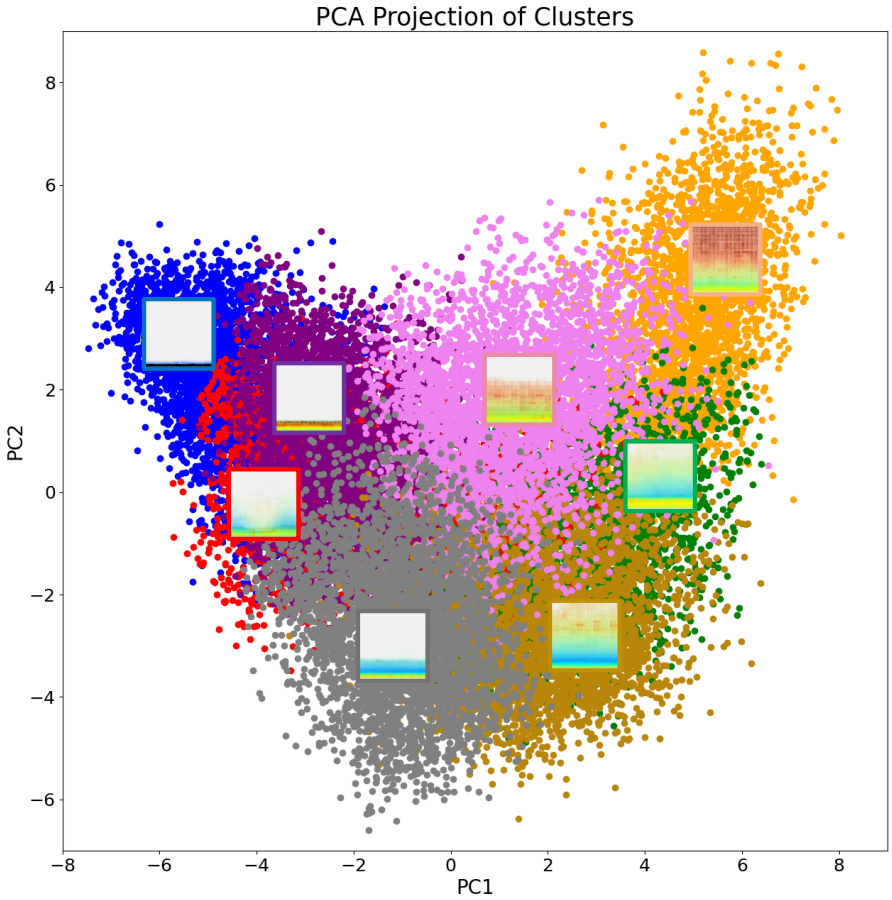


Figure 12: Visualization of clusters and centroids using the first two principal components.

Fig. 12 shows a 2-d visualization of the received clusters using principal component analysis (PCA) [8]. The first two PCs are used to plot the clusters. Also shown are the centroids for each cluster, visualized using the decoder part of the autoencoder. While the centroid images are kind of blurry, defined clusters can be made out. For example present in the blue, purple, and red clusters are phenomena that are at low altitudes. Wandering to the grey and pink clusters, the altitudes of the phenomena rise. In the top right corner, profiles with high backscattering values seem to dominate.

Fig. 13 is the same PCA visualization of the clusters. But this time some example segments (not the reconstructed centroids) and their position in the PC1-PC2 space are visualized, presenting a clearer picture of what the clusters might represent. Their respective cluster assignment can be recognized by the color of their borders.

The blue cluster groups very low-altitude phenomena like fog or precipitation, where barely anything above a very low height is visible. Next to it are the red and purple clusters. Both represent mid-altitude phenomena. The samples from the red cluster seem more cloudy and more prone to precipitation while the purple ones show more general aerosol patterns and fewer clouds. Comparing the positions of the pink and purple examples it is also apparent that the clusters do have a certain degree of overlap. This is to be expected since often the phenomena do have a smooth transition to each other or multiple phenomena do appear in the same window. The grey cluster appears to group segments that display mainly a clear sky and sometimes some aerosol. Going through the golden, green, and orange clusters, an interesting phenomenon can be observed: the amount of "brownish" color in the profiles seems to increase. This is a noise effect. Especially during noontime, the backscattering above clouds can be very high, resulting in the dark color. In future work, this

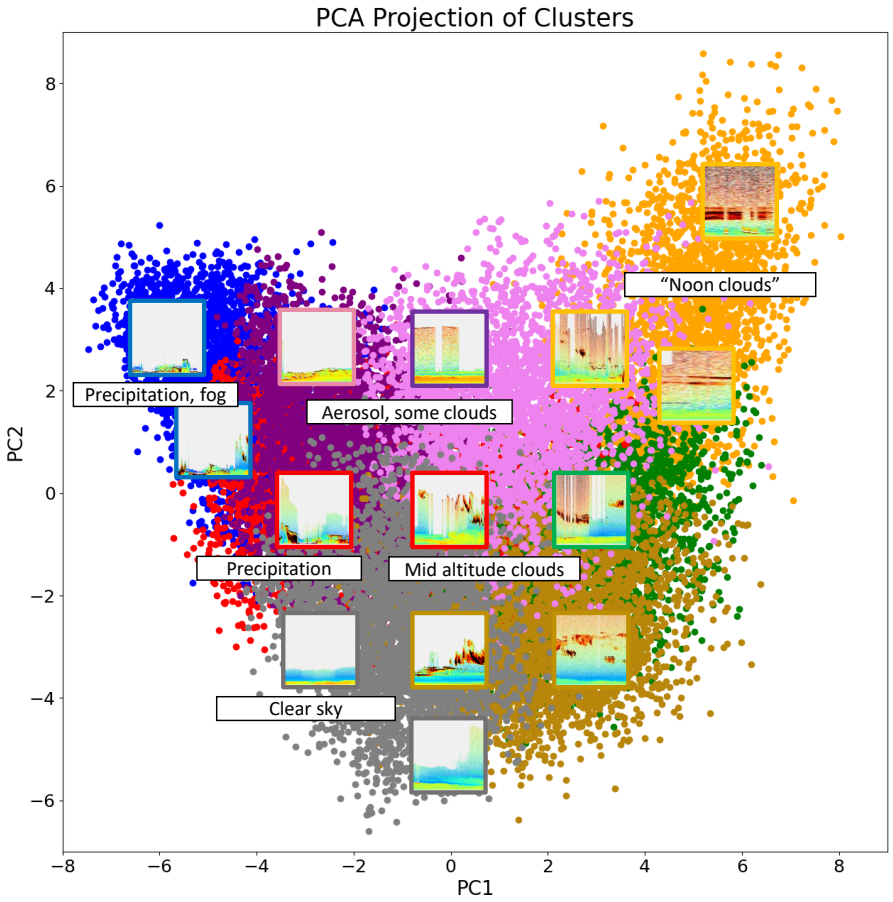


Figure 13: Example segments for several coordinates in the PC1 and PC2 projection space and interpretation thereof.

effect will be tried to be filtered out, since it bears no meaningful pattern and whether a phenomenon is occurring during noon time is encoded in the daytime encoding already.

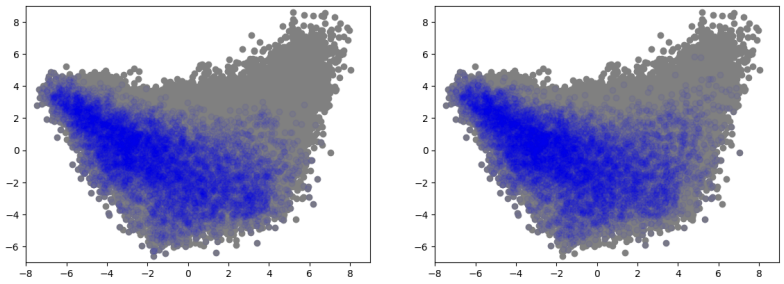
C.2 Analysis of time dependencies

Using the daytime and month encodings, an analysis of when certain phenomena occur can be performed. Fig. 14 visualizes subsets of the data that occur in different ranges of the day. Again, the "noon clouds" can be made out, looking at the (d) 9-12 am and (e) 12 am-3 pm plots. Apart from that, the detected phenomena seem to be quite distributed and not much dependent on daytime.

Fig. 15 shows subsets of the data depending on the month they occur. Here, it can be seen that the "noon clouds" are a phenomenon that takes place mainly in the summer. Furthermore, comparing for example (a) January and (b) July it is visible that the left edge of the point cloud is more prominent in winter than it is in summer. These are the clusters that represent phenomena like fog and precipitation. This could perhaps be attributed to (i) fog appearing more often in winter than in summer and (ii) dry periods during the summer (e.g., the summer in Germany in 2018).

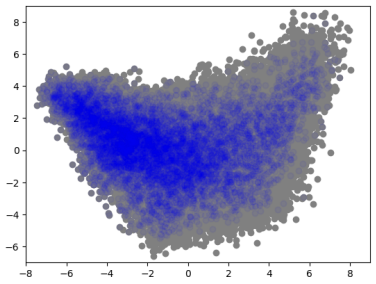
Fig. 16 shows subsets of data that each range in different durations. In (a) it is apparent that the "noon clouds" are often a comparatively short phenomenon, ranging from 0 to 3 hours. This may be due to the segmentation algorithm being very sensitive to the brownish color and resulting in over-segmentation. The mid-altitude clouds (center bottom) seem to be very prevalent in the 6 to 9 hours range (see (c)). All in all, the phenomena seem to be distributed widely among the clustering space. One also has to keep in mind that the detected durations are heavily dependent on the used segmentation algorithm and its parameters. Mid-altitude clouds being prevalent in the 6 to 9 hours

range could also indicate that some under-segmentation is taking place and that the actual single clouds are much smaller.

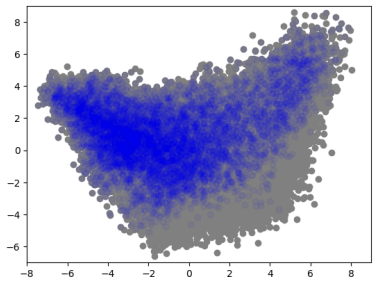


(a) 0-3 am

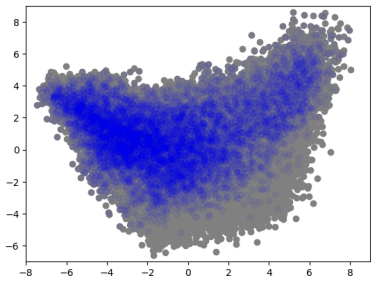
(b) 3-6 am



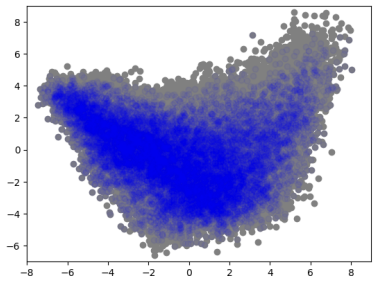
(c) 6-9 am



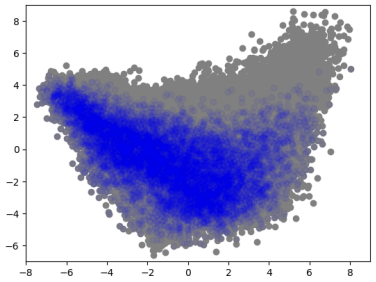
(d) 9-12 am



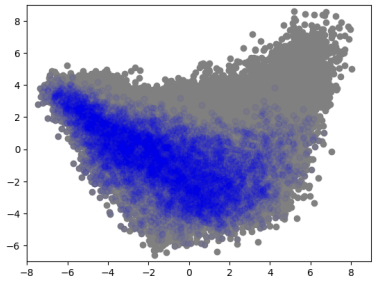
(c) 12 am-3 pm



(d) 3-6 pm

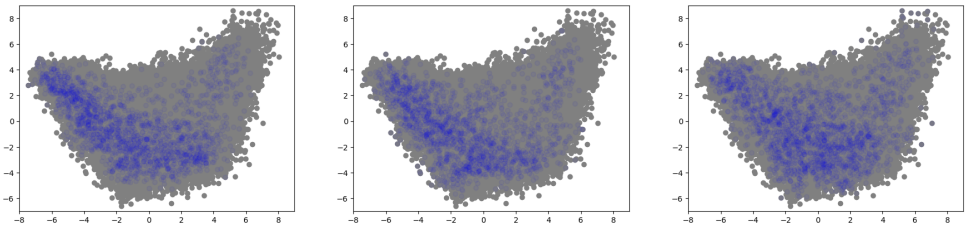


(c) 6-9 pm

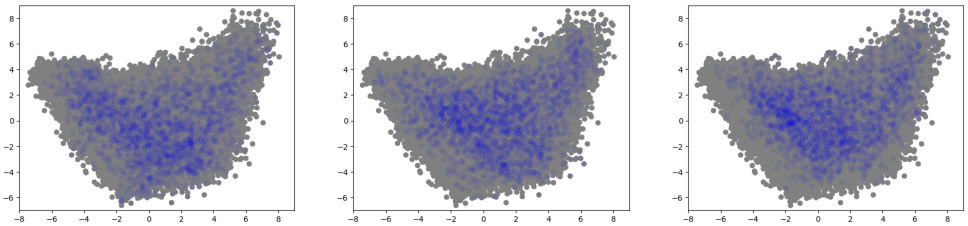


(d) 9-12 pm

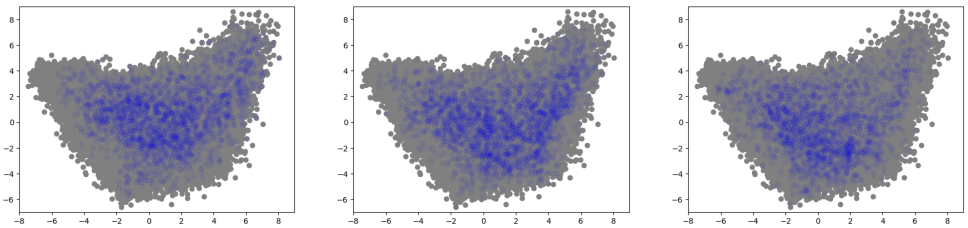
Figure 14: Position of phenomena depending on daytime they occur. x- and y-axes are PC1 and PC2.



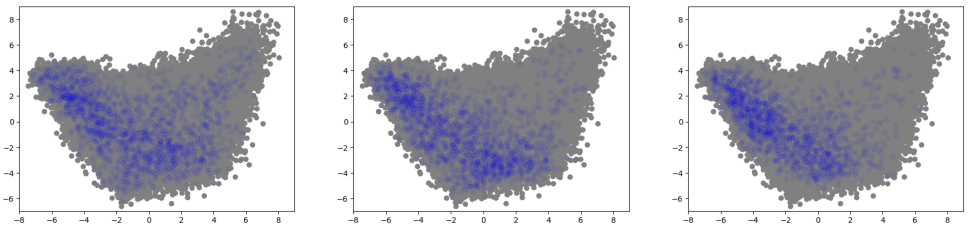
(a) January (b) February (c) March



(a) April (b) May (c) June

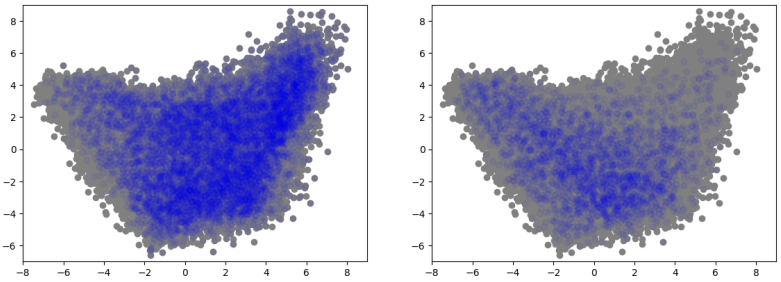


(a) July (b) August (c) September



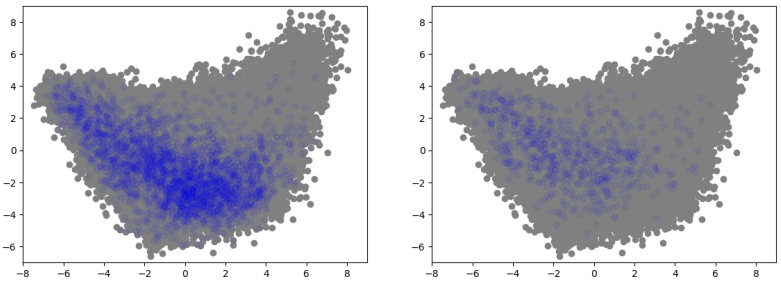
(a) October (b) November (c) December

Figure 15: Position of phenomena depending on the month they occur. x- and y-axes are PC1 and PC2.



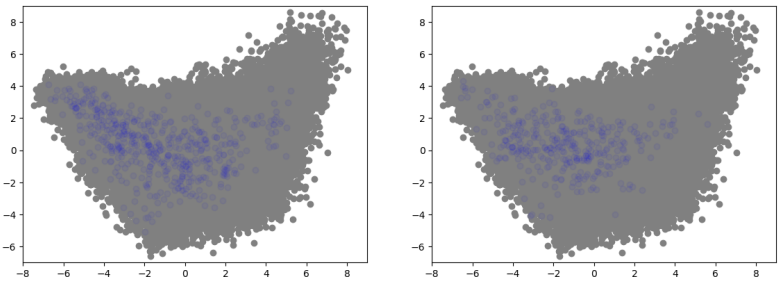
(a) 0-3 hours

(b) 3-6 hours



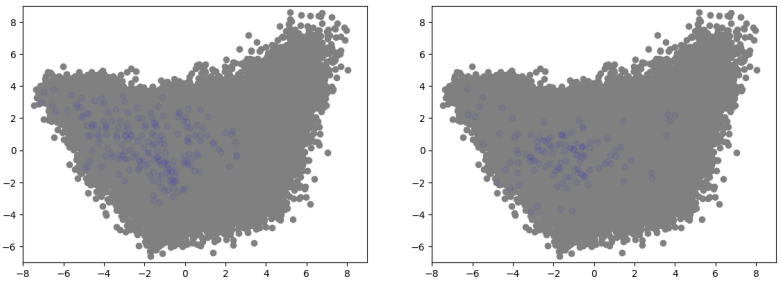
(c) 6-9 hours

(d) 9-12 hours



(c) 12-15 hours

(d) 15-18 hours



(c) 18-21 hours

(d) 21-14 hours

Figure 16: Position of phenomena depending on their duration. x- and y-axes are PC1 and PC2.

C.3 Implementation

The `scikit-learn` 1.0.2 implementation of k-means was used:
`sklearn.cluster.KMeans`.
The following parameters were applied: n_clusters=8, random_state=0.

The `scikit-learn` 1.0.2 implementation of PCA was used:
`sklearn.decomposition.PCA`.
The standard configuration was used.