DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Cloud Segmentation and Classification from All-Sky Images Using Deep Learning

Yann Fabel

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Cloud Segmentation and Classification from All-Sky Images Using Deep Learning

Wolkensegmentierung und -Klassifizierung von "All-Sky" Bildern mittels Deep Learning

Author:	Yann Fabel
Supervisor:	PD Dr. habil. Rudolph Triebel
Advisor:	Dr. Bijan Nouri
Submission Date:	July 14, 2020

I confirm that this master's thesis in informatics: robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, July 14, 2020 Yann Fabel

Acknowledgments

This master thesis was conducted at the Institute of Solar Research of the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V., DLR) within the department Qualification.

First of all, I would like to thank my supervisor PD Dr. habil. Rudolph Triebel who gave me the opportunity to write this thesis and supported me in professional matters. I highly appreciate his feedback, guidance and confidence in my research.

My sincere gratitude goes to my advisor Dr. Bijan Nouri. His continuous support, scientific advice and professional and personal encouragement were essential for the successful completion of this thesis. I am very thankful for the trust he put in me and my work and for the extensive discussion and feedback rounds.

I would also like to thank Dr. Stefan Wilbert and Niklas Blum for their counsel in meteorological questions and for the rich discussions. Furthermore, I would like to thank Wolfgang Reinalter and Lothar Keller for their constant assistance in IT matters.

My special thanks goes to my family and friends who have always supported me and made this work abroad possible. Finally, I would like to thank Luise for the endless patience, encouragement and support during my master's thesis.

Abstract

For transforming the energy sector towards renewable energies, solar power is regarded as one of the major resources. However, it is not uniformly available all the time, leading to fluctuations in power generation. Clouds have the highest impact on short-term temporal and spatial variability. Thus, forecasting solar irradiance strongly depends on current cloudiness conditions. As the share of solar energy in the electrical grid is increasing, so-called nowcasts (intra-minute to intra-hour forecasts) are beneficial for grid control and for reducing required storage capacities. Furthermore, the operation of concentrating solar power (CSP) plants can be optimized with high resolution spatial solar irradiance data.

A common nowcast approach is to analyze ground-based sky images from All-Sky Imagers. Clouds within these images are detected and tracked to estimate current and immediate future irradiance, whereas the accuracy of these forecasts depends primarily on the quality of pixel-level cloud recognition. State-of-the-art methods are commonly restricted to binary segmentation, distinguishing between cloudy and cloudless pixels. Thereby the optical properties of different cloud types are ignored. Also, most techniques rely on threshold-based detection showing difficulties under certain atmospheric conditions.

In this thesis, two deep learning approaches are presented to automatically determine cloud conditions. To identify cloudiness characteristics like a free sun disk, a multi-label classifier was implemented assigning respective labels to images. In addition, a segmentation model was developed, classifying images pixel-wise into three cloud types and cloud-free sky. For supervised training, a new dataset of 770 images was created containing ground truth labels and segmentation masks. Moreover, to take advantage of large amounts of raw data, self-supervised pretraining was applied. By defining suitable pretext tasks, representations of image data can be learned facilitating the distinction of cloud types. Two successful techniques were chosen for self-supervised learning: Inpainting-Superresolution and DeepCluster. Afterwards, the pretrained models were fine-tuned on the annotated dataset. To assess the effectiveness of self-supervision, a comparison with random initialization and pretrained ImageNet weights was conducted.

Evaluation shows that segmentation in particular benefits from self-supervised learning, improving accuracy and IoU about 3% points compared to ImageNet pretraining. The best segmentation model was also evaluated on binary segmentation. Achieving an overall accuracy of 95.15%, a state-of-the art Clear-Sky-Library (CSL) is outperformed significantly by over 7% points.

Kurzfassung

Für die Umstellung des Energiesektors auf erneuerbare Energien gilt die Solarenergie als eine der wichtigsten Ressourcen. Sie ist jedoch nicht immer einheitlich verfügbar, was zu Schwankungen in der Stromerzeugung führt. Die größte Wirkung auf kurzfristige zeitliche und räumliche Variabilität haben Wolken. Daher hängt die Vorhersage der Einstrahlung stark von der aktuellen Bewölkung ab. Da der Anteil von Sonnenenergie im Stromnetz zunimmt, sind Kürzestfrist-Vorhersagen (Vorhersagen innerhalb einer Minute/Stunde) vorteilhaft für die Netzregelung und um erforderliche Speicherkapazitäten zu verringern. Darüber hinaus kann der Betrieb konzentrierender Solarkraftwerke mit hochaufgelösten räumlichen Einstrahlungsdaten optimiert werden.

Ein üblicher Ansatz zur Kürzestfrist-Vorhersage ist die Auswertung bodengestützter Himmelsbilder von Äll-Sky-Imagern". Wolken innerhalb dieser Bilder werden erkannt und verfolgt, um die aktuelle und unmittelbare Einstrahlung abzuschätzen, wobei die Genauigkeit dieser Vorhersagen in erster Linie von der Qualität der Wolkenerkennung auf Pixelebene abhängt. Moderne Methoden beschränken sich häufig auf eine binäre Segmentierung, die wolkenfreie und Wolken-Pixel unterscheidet. Dabei werden optische Eigenschaften verschiedener Wolkentypen vernachlässigt. Außerdem sind die meisten Verfahren Schwellwert-basiert, was zu Schwierigkeiten unter gewissen atmosphärischen Bedingungen führt.

In dieser Arbeit werden zwei Deep Learning-Ansätze zur automatischen Bestimmung von Wolkenbedingungen vorgestellt. Um Bewölkungseigenschaften wie eine freie Sonnenscheibe zu identifizieren, wurde eine Multi-Label-Klassifizierung implementiert, die Bildern entsprechende Labels zuweist. Des Weiteren wurde ein Segmentierungsmodell entwickelt, das Bilder pixelgenau in drei Wolkentypen und wolkenfreien Himmel klassifiziert. Für das überwachte Lernen wurde ein neuer Datensatz mit 770 Bildern erstellt, der die zugrundeliegenden Labels und Segmentierungsmasken enthält. Um die großen Mengen verfügbarer Rohdaten zu nutzen, wurde zudem ein selbstüberwachtes Vortraining durchgeführt. Durch die Bestimmung geeigneter Vorwandaufgaben können Repräsentationen der Bilddaten gelernt werden, die die Unterscheidung von Wolkentypen vereinfacht. Zwei erfolgreiche Techniken wurden für selbstüberwachtes Lernen gewählt: Inpainting-Superresolution und DeepCluster. Anschließend wurden die vortrainierten Modelle auf dem gelabelten Datensatz fein justiert. Um die Wirksamkeit der Selbstüberwachung zu beurteilen, wurde ein Vergleich mit zufälliger Initialisierung und vortrainierten ImageNet-Gewichten durchgeführt.

Die Auswertung zeigt, dass insbesondere die Segmentierung von selbstüberwachtem Lernen profitiert und eine Verbesserung der Genauigkeit und der IoU von etwa 3%-Punkten im Vergleich zum ImageNet-Vortraining erzielt wird. Das beste Segmentierungsmodell wurde zudem für die binäre Segmentierung ausgewertet. Mit einer Genauigkeit von 95,15% übertrifft das Model eine moderne Clear-Sky-Library deutlich um über 7%-Punkte.

Contents

Acronyms

AM Air Mass. 13, 23

ANN Artificial Neural Network. 5, 6, 15

ASI All-Sky Image. 2–4, 9, 10, 12, 14, 15, 17, 18, 22, 25–29, 33, 36, 37, 40–42, 44–47, 49, 53, 54, 60–62, 64–66

BN Batch Normalization. 7, 29

CCSN Cirrus Cumulus Stratus Nimbus. 25

CNN Convolutional Neural Network. 6, 7, 9, 11, 12, 15, 16, 30, 33, 39, 41, 44, 61–63, 65, 74

CSL Clear Sky Library. 13–15, 17, 22, 41, 60–63, 65, 66, 74

CSP Concentrating Solar Power. 1, 17

DC DeepCluster. 34, 44, 46, 47, 49, 51, 54, 55, 58, 60, 61, 64

DLR Deutsches Zentrum für Luft- und Raumfahrt e.V.. 66

DNI Direct Normal Irradiance. 1, 27, 66

FC fully-connected. 8, 15, 39, 40, 46

FCN Fully Convolutional Networks. 8, 15, 16, 30

FN false negatives. 46, 51, 54

FP false positives. 46, 51, 54

GAN Generative Adversarial Network. 11, 37

GPU Graphical Processing Unit. 37, 44

HSI Hue-Saturation-Intensity. 13

HYTA Hybrid Thresholding Algorithm. 15, 25

ICNR Initialization to Convolution NN Resize. 30

IoU Intersection over Union. 41, 53–55, 58–61

IP-SR Inpainting-Superresolution. 34, 47, 54, 55, 64

kNN k-nearest neighbor. 12

LR learning rate. 35, 36, 39, 42, 44

MAE Mean Absolute Error. 39, 42

NWP Numerical Weather Prediction. 2

PCA Principal Component Analysis. 15, 40

PSA Plataforma Solar de Almeria. 17, 26, 27

PV photovoltaics. 1

PZA Pixel-Zenith-Angle. 3, 13, 14, 47

ReLU Rectified Linear Unit. 7, 8, 29, 34, 39

RGB Red-Blue-Green. 13, 40, 42

SPA Sun-Pixel-Angle. 13, 14

SVM Support Vector Machine. 12, 15, 16

SWIMCAT Singapore Whole-sky Imaging Categories. 25

SWIMSEG Singapore Whole-sky Imaging Segmentation. 25

SZA Solar-Zenith-Angle. 22, 24

TL Linke Turbidity. 3, 13, 22, 23, 44

TN true negatives. 46, 54

TP true positives. 46, 51, 54

VGG Visual Geometry Group. 7, 39

WMO World Meteorological Organization. 12, 16

1. Introduction

1.1. Motivation

Automatic detection and classification of clouds in the sky have several use cases. For instance, prevailing cloud types and coverage factors are crucial information for meteorology and climatology [1]. Until today, cloud observation is still mainly done by humans that regularly watch the sky and document their findings by hand. Besides high efforts for training professionals, the subjective nature of this task makes it hard to get uniform standards. An automated approach offers the possibility to extend observations and gather more data to further study the effects of clouds on global warming, for example.

Another yet increasingly important aspect of cloud observation is in terms of solar energy. Solar power plants utilize shortwave solar irradiance which has a high degree of variability. Seasonal and daily variation due to the sun-earth geometry and the earth's rotation can be easily taken into account by power plant operators. Intra-minute and intra-hour fluctuations however are mainly caused by clouds [2] which are much more challenging to predict. Other aerosols and atmospheric trace gases also affect solar irradiance but to a far smaller extent. Consequently, to predict short-term variation, accurate detection and type specification of clouds play a key role. So-called nowcasts, describing intra-hour forecasts, can be beneficial for optimized control in solar power plants and electricity grids, leading to more efficient exploitation of solar energy.

Especially for photovoltaics (PV) without battery storage, intra-hour variation results in high power ramps that cannot always be handled by electricity grids [3]. Even for larger grids, such fluctuations are considered problematic if the share of PV exceed 15% [4]. Nowcasting solar irradiance could improve grid stability in cooperation with smart control and trading mechanisms to distribute power over the grid. Moreover, it can help to reduce required storage capacities, as batteries could be utilized more efficiently [5], or to operate plants more flexibly [6].

Concentrating Solar Power (CSP) represent another technology of solar renewable energy. Solar fields using CSP consist of large mirror structures which focus solar irradiance on a receiver that contains a circulating heat transfer fluid. These thermo-hydraulic processes are less prone to large power ramps due to intrinsic inertia and can provide base load when combined with thermal energy storages. A potential benefit of irradiance nowcasts is to optimize the control of these facilities. Unlike PV, CSP use Direct Normal Irradiance (DNI), as concentration excludes most of the diffuse irradiance. Spatial variability of DNI within extensive solar fields poses complex control challenges [7] that aim to minimize cooling/overheating effects and harmful thermal stresses while maximizing heat flow [8].

Physical models for Numerical Weather Prediction (NWP) and satellite observations are two established methods used for irradiance forecasts. NWPs can provide forecasts up to several days ahead in hourly and coarse spatial resolution [9]. Geostationary satellites enable forecasts up to 9 hours ahead [10] with typical spatial resolutions of 2-10 km and temporal resolutions of 15 minutes [11]. Most advanced systems, such as GOES-R, reach spatial resolutions of 0.5 km^2 and temporal resolution of 5 minutes [12].

Neither NWP nor satellite based forecasting reach the requirements for spatial and temporal resolution for effective control optimization of solar power plants, storage facilities or electrical grids.

Ground-based cameras are an alternative that can provide both, sub-minute temporal resolution and spatial resolution of less than 50 m. Several devices have been presented in literature, like the Whole-Sky Imager [13], the Total-Sky Imager [14], or the All-Sky Imager [15]. They all represent cameras with fish-eye lenses that can observe almost the entire hemisphere. In this work, the term All-Sky Imager is used and further details of the hardware can be found in section 3.1. Using All-Sky Imagers, various nowcasting systems [16, 17, 18, 19] have been developed which use sky images to detect, geolocate and track clouds. Since the detection of clouds is the first step, the quality of the forecast depends strongly on these results. In addition, classifying detected clouds enables more precise determination of their radiative effects. Furthermore, automated classification offers valuable information to further study optical properties of clouds which is important for current climate research and meteorology.

1.2. Objectives and Challenges

This thesis aims to develop two new methods to detect and classify clouds from ground-based observations using All-Sky Imagers. Both are based on deep learning techniques where a computational model learns to classify All-Sky Images (ASIs) from data. The first constitutes a multi-label classification. It deals with recognizing image properties by assigning multiple labels to an image. The other performs pixel-wise classification, allocating a unique class to each image pixel, also known as semantic segmentation. The goal is to categorize pixels into regions or groups that share optical similarities. While the former provides information regarding global characteristics, the latter allows to exactly locate clouds within the image. Semantic segmentation can also be regarded as a step further to finer inference in scene understanding [20]. The image-wise labels are defined to obtain information about current cloudiness conditions and to determine which cloud types are present in the image. Clouds are divided into three classes: *low-*, *mid-* and *high-layer* to account for varying optical properties and different dynamics. The segmentation model aims to classify each pixel into one of these classes (or cloudless sky). Developing both methods also allows better understanding of the segmentation results. Challenges in classification apply for segmentation, too. Hence, it can be determined whether there are difficulties in identifying the correct cloud classes or in localizing them accurately within the image.

Corresponding to [21], clouds are defined by the amount of water in form of droplets or ice

crystals and their visibility. This implies a differentiation to aerosols in general, as aerosols can be composed of any dispersion of solid and liquid particles independent of visibility or density. Still, clouds and aerosols describe the same phenomenon and it is not always easy to separate one from the other.

Another challenge in cloud detection is the so-called twilight zone. It is described as *"a belt of forming and evaporating cloud fragments and hydrated aerosols extending tens of kilometers from the clouds into the so-called cloud-free zone"* [22]. The absence of sharp boundaries in many cloud types makes accurate segmentation particularly difficult. Even for human observers distinction is not always trivial and subjective perception cannot be completely avoided.

Apart from variable structures of clouds in general, their appearances are largely effected by illumination and atmospheric conditions. The same cloud can change its visual characteristics entirely when passing in front of the sun, sometimes leading to large overexposed regions in the image. Simultaneously, the circumsolar area (image region in close proximity to the sun disk) is of particular interest for solar energy applications. Clouds that are close to the sun disk from a certain perspective cast their shadow directly on the observer, leading to locally altered irradiance. Atmospheric turbidity, measured by the so called Linke Turbidity (TL) [23] also influence perception of clouds. It describes the reduction of solar irradiance as a multiplier of clear and dry atmospheres.

The fish-eye lens of an All-Sky Imager poses another challenge. Due to small viewing angles, the resolution in the image decreases with increasing Pixel-Zenith-Angle (PZA). Thus, clouds close to the horizon can be very extensive, but at the same time only take up a fraction of the image. This applies in particular for high clouds, as they could be dozens of kilometers away. Additionally, absolute changes in color channels are generally smaller at the edges of ASIs due to darkening caused by the lens (vignetting) [24] and atmospheric effects.

Furthermore, distinct cloud types share many visual similarities which makes them hard to distinguish. Especially multi-layer situations impede segmentation when clouds of distinct classes overlap. Such cases, which are not rare, represent the most difficult conditions.

Because of the complexity of these conditions, most state-of-the-art methods apply binary segmentation. Hence, cloud types are not distinguished and multi-layer conditions are neglected. This work presents one of the first approaches to segment ASIs into multiple cloud classes. A fundamental part is training the models in a supervised manner on a versatile dataset that includes complex multi-layer situations. The goal is to provide highly resolved automated cloud classification, coverage determination and distinction of multiple layers. Additionally, I evaluate the effectiveness of self-supervised learning by applying pretraining on a large unlabeled dataset. Two successful techniques are implemented and evaluated by comparing them with ImageNet [25] pretraining and random initialization.

The remainder of this thesis is organized as follows. In chapter 2, state-of-the art techniques in deep learning as well as current methods for classifying and segmenting ASIs are presented. The utilized datasets for supervised and self-supervised learning are introduced in chapter 3. A detailed description of the developed models can be found in chapter 4. Presentation and discussion of experimental results are covered in chapter 5. Finally, a conclusion of this work is given in chapter 6 as well as a brief overview on future work.

2. Related work

In the following sections, an overview of current deep learning techniques and state-of-the-art ground-based cloud observation is given. Starting with general concepts of machine learning, recent publications in deep learning are presented afterwards. The last part deals with various methods described in the literature for detecting clouds in ASIs.

2.1. Machine learning in image processing

Machine learning and the subcategory deep learning are very current research areas that have a broad range of applications. Particularly in computer vision, data-driven approaches make up a majority of the latest publications. This constantly results in new insights offering new possibilities. In this section, I describe established methods as well as recent findings that helped to develop the models. A major focus is set on learning data representations with unsupervised pretraining to tackle the problem of limited annotated datasets.

2.1.1. Definition and terminology of machine learning

A frequent challenge in developing technical solutions for apparently easy problems, like seeing clouds in the sky, is the difficulty and complexity of describing such problems mathematically. Machine learning, as a technique of artificial intelligence, aims to tackle such problems by acquiring knowledge from data.

Usually there are two phases in machine learning. The first one is the learning process in which a mathematical model adapts its internal parameters by searching for patterns in the training data. The second one is using the trained model to make predictions about future unknown data samples.

To achieve good generalization, the quality and quantity of data is essential. Otherwise, if there is only little noisy data, overfitting may occur. Thus the model learned to recognize patterns that only exist in the particular training set.

As raw data is often represented in high-dimensional space but only specific parts are needed to solve the task, many approaches apply preprocessing in order to find desired patterns. This procedure is also known as feature extraction which transforms the input into more useful representations.

Generally, three kinds of machine learning approaches are distinguished: supervised, unsupervised and reinforcement learning. For supervised problems, training data contains the input as well as a so called ground truth, a target vector that the model should output given the input. When the target consists of continuous variables, it is called regression. If the input shall be assigned a value from a finite set, it is a classification problem.

In unsupervised learning, there is no ground truth but the data consists only of the input. Common tasks are discovering groups of similar samples (clustering) or determining data distribution (density estimation). Reinforcement learning deals with finding appropriate actions for given inputs by applying a reward system that encourages desired behavior.

Further details on machine learning in general and the mentioned concepts can be found in [26, 27, 28].

2.1.2. Learning visual representations with deep neural networks

Deep learning is a machine learning technique and a form of representation learning. The characteristic of learning representations from data is that it does not require carefully designed feature extractors to perform pattern recognition. Instead, in deep learning a computational model learns to extract important features from raw data so that it is represented in a distinctive way. The goal is to learn from experience and develop an understanding of the world which is based on hierarchical concepts [27].

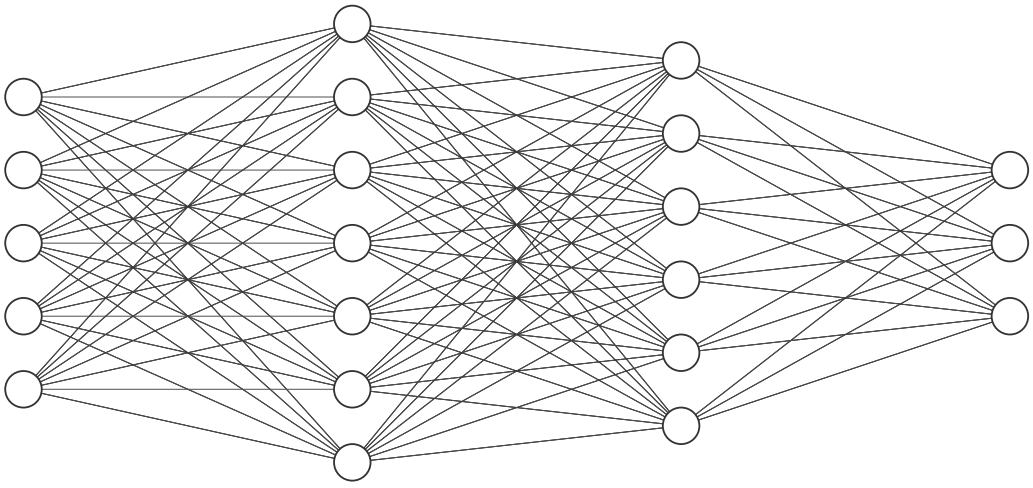


Figure 2.1.: Example of 3 layer fully-connected ANN. Each neuron (circle) is connected to all neurons of the successive layer.

One of the key attributes of these computational models is the composition of multiple interconnected processing layers. For each layer, the input data is transformed further, leading to more abstract representations and building a hierarchically structure of information processing. Probably one of the most prominent models in deep learning are Artificial Neural Networks (ANNs). Composed of multiple layers each containing various non-linear modules, often referred to as “neurons”, ANNs can represent any mathematical function [29]. An example of an ANN is shown in figure 2.1. The key technique to learn the respective input-output relation is called backpropagation. Thereby, the internal parameters, also known as weights, are adjusted to produce better representations and thus a more precise mapping.

In supervised learning, this is achieved by computing the error between the network output and the actual target value with a loss function. Computing the gradients of this function with respect to the weights, they can be adjusted to decrease the error.

Due to the increasing availability of computational power and the progress in effectively training neural networks, deep learning methods gained a lot of attention in the past decade, in particular since those methods have outperformed other techniques in applications like image [30] or speech recognition [31] by far. Like other natural signals, images can be decomposed hierarchically. This is why deep learning works so well in image processing. For instance, an image can be seen as a composition of edges creating motifs which make up parts and finally objects [29]. Using techniques as described in [32], this hierarchy can even be visualized as shown in figure 2.2.

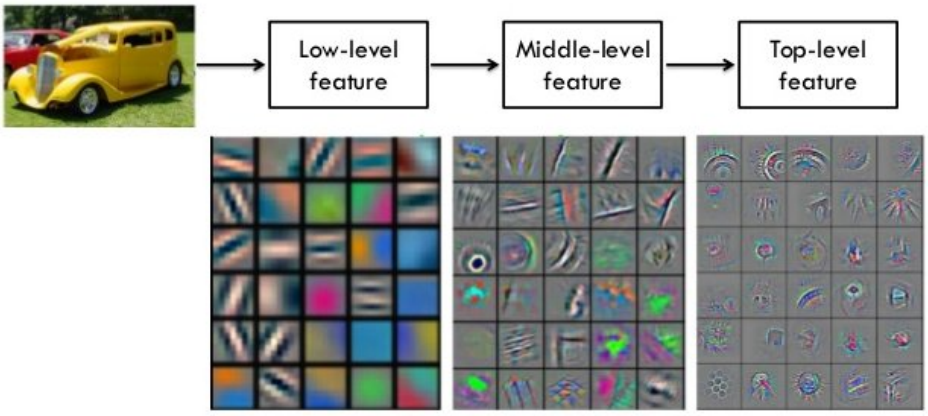


Figure 2.2.: Visualizing hierarchy as described in [32]. Image obtained from [33]

.

A famous example for the success of deep learning in image recognition is the winner of the ImageNet challenge [25] in 2012, AlexNet [30]. The task was to classify images into one of 1000 classes. For training, the ImageNet dataset contained 1.2 million labeled images at that point. Beating the second best candidate by over 10 percent in the top-5 test error rate, AlexNet marks a cornerstone in the development of deep learning and lead to increased popularity of a specific type of ANNs for image processing: the Convolutional Neural Network (CNN).

2.1.3. Convolutional neural networks

A CNN is composed of different type of layers, the most significant one being a convolutional layer. A so-called kernel, or filter, consisting of a set of weights and structured as a 3-dimensional tensor is shifted over different input regions by a predefined stride, applying matrix multiplication at each position (see figure 2.3). Stacking the results of several filters per layer forms a new output tensor.

The concept of sharing weights with different parts of the input reduces the number of trainable parameters and enables detection of local patterns independent of spatial location.

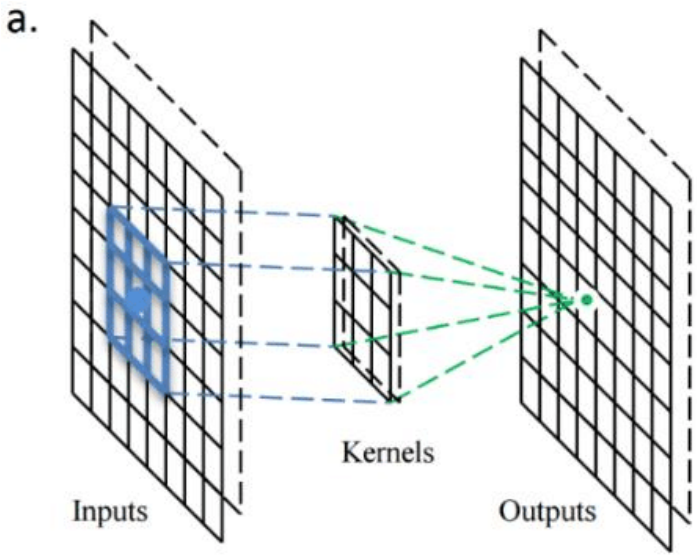


Figure 2.3.: Example of convolutional layer (adapted from [34])

.

Subsequently, the output tensor of the convolutional layer is transformed element-wise by a non-linearity, typically a Rectified Linear Unit (ReLU) [35].

Another commonly applied layer is the pooling layer which merges features that are semantically similar. Due to this setup of concatenating multiple stages of convolutional layer, non-linearity, and pooling layer, a CNN can learn to extract high-level features composed of low-level ones. This enables detection of features independent of their location, simultaneously being robust to some degree of variation.

A very successful architecture is the VGG net [36], named after the team that won the ImageNet challenge 2014. They used more layers than other network architectures at the time and small filter sizes of 3x3 for all of them.

However, very deep networks appeared to be hard to train due to the vanishing gradient problem. As weight gradients are computed by back-propagating the error through the network, repeated multiplication of preceding gradients may lead to infinitely small gradients. Consequently, the weights of early layers can not be adjusted and the network performance saturates.

It appeared that the performance of a trained network strongly depends on the initialization of its weights. Thus, several studies examined normalization approaches for weight initialization [37, 38, 39, 40]. Furthermore, normalizing layer inputs by integrating so-called Batch Normalization (BN) layers [41] into the architecture have become common practice. Another solution to prevent vanishing gradients even for very deep networks was proposed in 2016, describing deep residual learning for CNNs and presenting the architecture of ResNets [42]. The main characteristic in ResNets is an identity "shortcut connection". Here, the output of one layer x is directly forwarded to a later layer, skipping some layers in between, and then added to the output of the preceding layer $\mathcal{F}(x)$ (see figure 2.4).

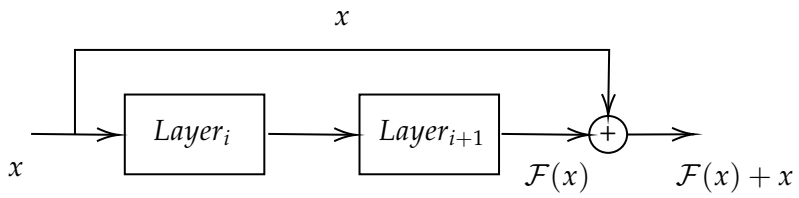


Figure 2.4.: Residual block in ResNets

The authors hypothesize that it is easier to optimize a residual mapping $\mathcal{F}(x) + x$ than the original unreferenced mapping since the gradient in such a residual block is now enlarged by the derivation of the identity. Referring to empirical studies on ResNets [42], and the fact that this architecture won the 2015 ImageNet challenge [25], they indeed achieve better accuracies with deeper networks and avoid the problem of vanishing gradients.

Regarding pixel-wise classification of images (semantic segmentation) another architecture became widely popular. Fully Convolutional Networks (FCN) [43] use deconvolutions, or backward convolutions, for upsampling dense feature tensors back to the input image size predicting a class for each pixel. As for classification models, multiple stages of convolution, pooling and non-linearities produce abstract representations. Instead of a subsequent fully-connected (FC) network, these representations are processed by deconvolutional filters, which can be seen as convolutions with fractional input stride. Stacked together with activation functions, like ReLU, deconvolution allows learnable, non-linear upsampling. Furthermore, to combine global semantic information with local appearances, [43] also introduces skip connections from earlier layers with finer strides to the final output layer. Hence, higher-resolution features of the convolutional part are fused with the predictions of the last upsampling layer. Applying another convolution to the combined result, the network can learn to predict more precisely.

This idea is even extended in [44] describing an architecture originally applied for segmentation in medical images, called U-Net. Instead of only combining the last upsampling layer with the output from earlier convolutions, the U-Net architecture has a symmetric u-shaped structure of multiple skip connections between the downsampling and upsampling part of the FCN. The output from various layers during feature extraction (downsampling) is concatenated to the input of the corresponding layers in the upsampling process, sharing local information of the respective resolution and thus generating more precise segmentation masks.

2.1.4. Limits of supervised learning

The models described previously were all trained and evaluated in a supervised manner. There is a variety of labeled datasets commonly used for benchmarking new machine learning techniques or models, like ImageNet [25] for image classification, or PASCAL VOC [45] and COCO [46] for segmentation. Thus, for all images in the dataset, there exists a respective ground truth determining the target that should be predicted by the model. For image classification, the ground truth is the class corresponding to the depicted object, usually

represented by an integer. In segmentation tasks each pixel of the input image is classified separately. When comparing the model's prediction with the corresponding ground truth using a specified loss function, the error can be evaluated and the gradients are computed using the backpropagation algorithm. Typically, stochastic gradient descent or some variant thereof is applied for updating the model. The model is shown a few examples of the dataset (a mini-batch), computes the output, errors, and gradients and then averages the gradients for a single update step.

However as mentioned before, a large amount of training samples is required to train a model that generalizes well to unseen data. Otherwise, the model may overfit, memorizing the ground truth of the training set instead of learning general distinctive features. To overcome this issue, it is important to cover different perspectives and possible appearances of a given class, such that the model can learn to create visual representations, required for determining the correct class. In practice, the acquisition of labeled datasets, i.e. data with corresponding ground truth, is time-consuming and sometimes even infeasible for millions of images. Therefore, application specific datasets are often magnitudes smaller than popular benchmarking datasets. Especially if the effort of annotation is high, as for segmentation, datasets are often limited to a few dozen to hundred samples. For instance, in [44] only 30 images with associated ground truth were used to train the U-Net. To deal with this problem, various techniques have been developed to prevent overfitting and train models that generalize well which are going to be discussed next.

2.1.5. Data augmentation and transfer learning

Data augmentation is a common technique to artificially enlarge an existing dataset. Either by randomly warping data samples during the training phase, e.g. applying affine transformations to the input, or by synthetically creating new instances and adding them to the training set which is called oversampling [47]. Figure 2.5 shows rotations/mirroring of an ASI as exemplary affine transformation.

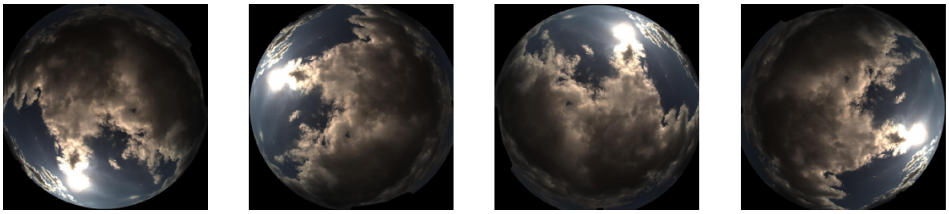


Figure 2.5.: Example of image rotation/mirroring as data augmentation method for ASIs

.

Another approach frequently applied for CNNs when data is limited, is transfer learning. A network is pretrained on a big dataset, like ImageNet, and the weights of the convolutional layers are used for initialization in the application task. Transfer learning can be applied to any vision task like classification, object detection or segmentation, provided that the downsampling architecture of the CNN is the same. Many image datasets share low- and

mid-level features which can be learned better on big datasets [48]. This is because of the hierarchical composition of image features leading to very general ones in early layers which become more and more specific in deeper layers (see figure 2.2). Even on dissimilar datasets, transferred weights usually show some benefit compared to random initialization such that it has become common practice to start training with pretrained weights. However, recent publications on transfer learning question the actual benefit of pretraining on datasets like ImageNet. In [49], it is shown that models with randomly initialized weights can reach competitive results on the COCO dataset [46] compared to the pretrained ImageNet counterparts. Regarding medical imaging, where image data is very different to ImageNet, it is concluded that only few pretrained filters contribute to useful feature extraction and the rest are too specialized from the original task [50]. Like in medical imaging, ASIs differ very much from datasets like ImageNet. For example, images often require higher resolutions in order to recognize the cloud type. Moreover, there is not necessarily a clear or unique target class within an image and there are considerably less classes overall. Therefore, recently another approach has gained a lot of attention in the computer vision community that allows to learn useful representations even if labeled data is not available.

Unsupervised and self-supervised learning

Unsupervised learning in general does not require a ground truth but it aims to discover underlying structures and patterns in the data. Apart from clustering or dimensionality reduction, learning representations with unlabeled data is a common task in unsupervised learning that has been studied intensively since the breakthrough of deep learning. The related approach of self-supervised learning [51] is a form of unsupervised learning. Within a so-called pretext task, the model learns visual representations in a supervised manner, but without any human-annotated labels which is why it is considered as unsupervised. This is achieved by extracting or generating the ground truth automatically from the image data. In the second part, the downstream task, the learned features are transferred to the model for the target application and the model is fine-tuned on a small dataset. The idea of self-supervision is to define a problem the model can only solve by learning important visual representations which are also useful for the actual target task. As the distinction between unsupervised and self-supervised methods is informal in the existing literature, in this work the more specific term self-supervised is used.

There are different approaches of visual representation learning that can be applied as pretext tasks. A survey on self-supervised learning [52] subdivides those methods into generation-based, context-based, free semantic label-based, and cross-modal based.

For generation-based methods, the auto-encoder is a well-known example. An auto-encoder consists of an encoding and a decoding part connected by a bottleneck layer. During the encoding, the input data is compressed into a latent-space representation and the decoder tries to reconstruct the input. A difference measure called reconstruction loss is then used for updating the model weights through backpropagation. The drawback of standard auto-encoding is that the model learns to reconstruct input noise as well. Vincent et al. presented a variant of stacked denoising auto-encoders for feature learning with deep networks. Later

CNNs were also used in terms of auto-encoding [54].

Another type of generative models that can be applied for unsupervised representation learning is the Generative Adversarial Network (GAN) [55]. The main characteristic of GANs is the competition of two networks trying to beat each other, like in a zero-sum game. While the generative part creates an output which should be similar to some given data distribution, the adversarial part aims to distinguish between the generated output and the true data. Although GANs are still relatively new, there are already a lot of examples where they were applied successfully. A famous version is the Deep Convolutional GAN (DCGAN) released in 2015 by Radford et al. who explicitly used GANs for representation learning. They trained one CNN to generate artificial images from a uniform distribution and another CNN to distinguish between artificially created and real images. The two networks improve each other as they try to get better in solving their respective task, the generative model by creating more realistic images, and the adversarial one by detecting finer details for discrimination. In another work, a GAN is used for predicting missing parts of an image [57], also known as inpainting. Here, the input is not drawn randomly from some distribution, but real images are corrupted by superimposing black boxes onto them. Hence, the model needs to learn to interpret the context by looking at surrounding areas of the missing parts in order to fill the boxes reasonably. Apart from creating artificial images or parts of it, GANs also have proven to be effective for superresolution [58]. Due to a perceptual loss, comprising pixel and adversarial loss, the model is forced to learn fine textures of a depicted object to produce realistic high-resolution images. The idea of including a perceptual loss for superresolution was also pursued in [59], introducing an extra loss from a trained and fixed convolutional *loss network* [59]. It is computed by measuring differences in feature maps at different layers of the fixed network between target and predicted images. As a result, when textural features get lost during upscaling, the discrepancy in feature maps leads to a higher loss and eventually pushing the model to consider these features.

Context-based pretext tasks aim to learn context similarities or spatial/temporal structures. Spatial relation methods include predicting relative positions of two image patches [60] and solving jigsaw puzzles [61]. When video data is available, temporal relation can be learned by predicting the correct sequence from a randomized set of images obtained from a video [62]. A method which outperformed other state-of-the-art methods when applying learned representations directly in benchmarks for classification, detection or segmentation tasks was presented in [63]. It is based on learning image similarities by alternately clustering CNN features and training the CNN to solve a classification problem.As clustering technique, the k-means algorithm is applied which groups input vectors to a predefined number of clusters by calculating a distance measure between all samples. Next, the images are assigned pseudo-labels which correspond to the clusters that the CNN outputs were previously allocated to. Using these pseudo-labels, the CNN is trained to classify the images leading to updating its internal parameters and thus also to new features. Repeating this procedure the model can learn to classify images based on visual similarities.

2.2. Current techniques for ground-based cloud imaging

This section deals with the literature on classification and segmentation of clouds in ASI. Most of the times, previous studies have examined these tasks separately due to inherent complexity. Apart from few recent publications, cloud segmentation generally focused on solving a binary problem, distinguishing clear sky from cloud pixels. Starting with reviewing current classification methods, it is continued with those for binary segmentation and concluded with studies that address a more fine-grained segmentation.

2.2.1. Cloud classification

In general classification tasks, the goal is to assign a given input to a category that is part of a predefined set. As machine learning methods encompass suitable and effective tools for categorizing data into groups of specified characteristics, most approaches in cloud classification are learning-based. Typically, spectral, textural or structural features are extracted from images and fed into a classifier. Heinle et al. [64] proposed a k-nearest neighbor (kNN) classifier, assigning ASIs into one of seven classes that represent different sky conditions. The classes are defined by merging some of the official cloud types that have similar visual characteristics. Since multi-layer conditions occur frequently, images for training were selected manually to leave out ambiguous situations and show a clear dominant cloud type. With a focus on combining textural and structural features, Zhuo et al. [65] applies a Support Vector Machine (SVM) to classify feature vectors. The class definitions are adopted from [64], though one cloud type was omitted. Unlike in [64], only parts of an ASI, so-called sky patches, were classified to prevent ambiguous conditions. More recently, deep learning methods were tested, too. In [66], a CNN based model was presented, called DeepCloud. First, features are extracted by a CNN. Then they are processed to obtain more distinctive representations by applying discriminative local pattern mining and Fisher vector encoding [67]. Finally, classification is performed with a SVM. The dataset is the same as in [65], containing 1231 images, but with a more precise classification of nine cloud types as defined by the World Meteorological Organization (WMO) [68]. A regular CNN for cloud classification was presented in [69]. It is based on AlexNet [30] and achieved very good results on their own dataset. They also used official cloud generas for classification but added the class of contrails caused by aircrafts, resulting in 10 different classes.

It should be mentioned that comparisons are not expressive when different methods are not evaluated on the same dataset. Depending on the selection of images, there are easier or more difficult conditions for classifying clouds. For different observation sites, turbidity may vary substantially and complex multi-layer situations may occur more or less frequently. To my knowledge, the only current benchmark for classification was performed by Ye et al. in [66], comparing also the methods from [64] and [65].

2.2.2. Cloud segmentation

Automatic cloud segmentation from ground-based camera images has been studied intensively the last two decades. Sometimes also referred to as cloud detection, in this work the term segmentation is utilized for pixel-level classification, whereas detection indicates recognition independent of localization. Several threshold methods on color space information were developed to classify pixels into sky or cloud. Within the Red-Blue-Green (RGB) color space, the ratio [70, 71] or difference [64] between the red and blue color channel is a common measure. Due to increased scattering of blue wavelengths (Rayleigh scattering) in the atmosphere under clear sky conditions, the sky appears blue and the red-blue ratio (or difference) is smaller. For clouds however, Mie scattering is dominant leading to white or gray appearances. Other works propose to include the green channel as well [72] or transform the image into other color spaces, like HSI, and examine saturation [73, 74]. Furthermore, super-pixel segmentation has been investigated in [75]. Instead of classifying each pixel, an accumulation of pixels with similar textures and brightness are combined to super-pixels which are classified as a whole. Graph models outline another strategy. Each pixel [76] or super-pixel [77] is considered as a node connected to neighboring (super-)pixels by weighted edges. Taking some initial nodes that were classified with high confidence, the rest of the image is classified by evaluating a cost function for each node which depends on the edges providing information from surrounding nodes.

A problem with fixed thresholds is that they only work under certain conditions. For example, the RB ratio strongly depends on the relative position of individual pixels to the sun or to the horizon, in particular for atmospheric conditions with high amounts of aerosols. To address this challenge, Clear Sky Libraries (CSL) [16, 78, 79, 80, 81] were introduced. They contain historical RGB data acquired from clear sky conditions. By comparing a clear sky reference image with the image of interest, difficult conditions around the sun become easier to segment. For validating my deep learning approach, a CSL as described in [81] is also applied for binary segmentation on the same dataset in section 5.3.3. The CSL stores RGB values corresponding to Sun-Pixel-Angle (SPA), PZA, Air Mass (AM), and TL. SPA and PZA describe the relative position of pixels to the sun and the zenith (see figure 2.6). AM is a measure for the relative path length of solar rays through the atmosphere and TL specifies the fraction of incoming solar irradiance compared to ideal, clear and dry atmospheres.

The CSL consist of multiple layers, each corresponding to a specific AM and TL. Within a layer, RGB values from clear sky conditions are stored with respect to SPA and PZA. To distinguish between cloud- and sky-pixels, a clear sky image is generated from the layer that reflects current TL and AM best. Color features from the target image, like RB ratio, are then compared with the ones from the reference image. Detection is determined by multiple thresholds depending on SPA, PZA and the current weather situation which is estimated by considering recent fluctuations in irradiance measurements. An illustration of the workflow of the CSL is depicted in figure 2.7.

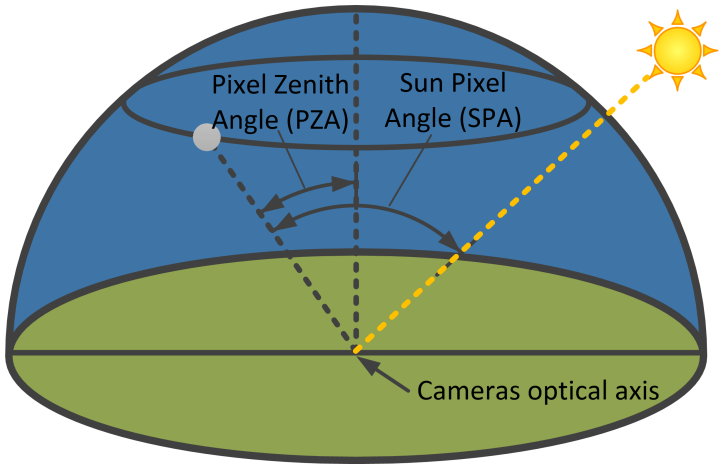


Figure 2.6.: Illustration of PZA and SPA for ASIs.

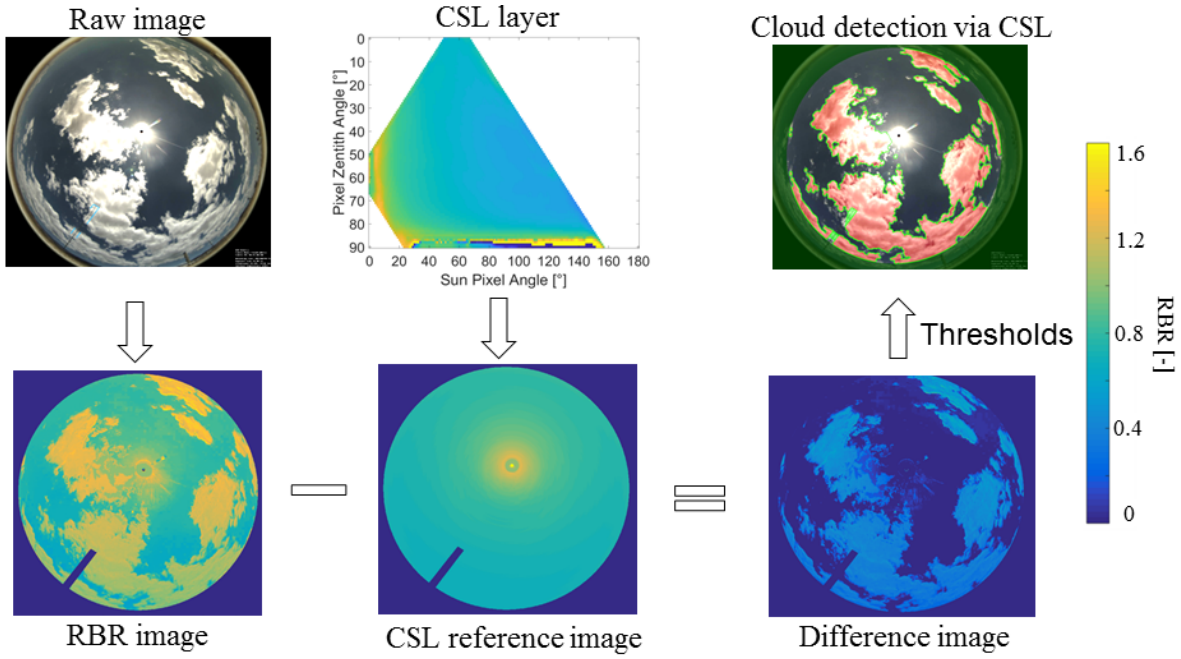


Figure 2.7.: Illustration of the working principle of a CSL (graphic adopted from [81]).

Over the past years, also some machine learning-based methods for cloud segmentation were presented. A systematic study on color spaces, applying Principal Component Analysis (PCA) and fuzzy clustering to identify the most relevant color components for cloud segmentation is applied in [82]. In another work [83], Dev et al. review machine learning techniques for ground-based image analysis. Particularly a set of feature extraction methods like dimensionality reduction, sparse representation and classic image feature extractors are discussed within the context of segmentation, classification and denoising. One of the first applications of neural networks for ground-based cloud images was published in [84]. In this study, a shallow FC ANN is compared to a SVM, both showing significant performance boost over a threshold-based method used for validation. Although this dataset contains images from different conditions of cloudiness, the models are only trained and tested on extracted pixels from 15 images in total. Later, multiple supervised learning methods, like random-forest, SVM, and Bayesian classifiers as well as combinations of multiple classifiers were examined in [85]. For training, feature vectors are constructed by concatenating values of different color spaces as well as the RB ratio for each pixel. Furthermore, color components of local patches are added to create multiple feature vectors for each pixel in order to integrate information from neighboring pixels. Here, the classifiers were trained on 250 images with the best model achieving an average accuracy of about 90% for 10-fold cross validation.

As deep learning achieved outstanding performance improvements in other computer vision tasks, there has been increased interest in CNNs for cloud segmentation from ASI. Most recently, three academic papers were published tackling the issue of cloud segmentation with CNNs [86, 87, 88]. All of them follow a purely supervised approach with an encoder-decoder architecture based on the original FCN [43]. The model presented in [87], called SegCloud, was trained and tested on 340 and 60 annotated all-sky images respectively and outperformed a R/B threshold based approach by a large margin. Song et al.[88] conducted an extensive study of popular deep learning segmentation architectures by applying them on a dataset of sky patches [89].

Since there exists a variety of different approaches for cloud segmentation, benchmarks are an important tool to compare them with each other and to identify strengths and weaknesses. This is in particular because most approaches use individual datasets which have different characteristics. While some datasets consist of sky patches, intentionally omitting areas of the sun to prevent overexposure, others comprise ASIs from different devices. A comparison of various methods was conducted on two datasets of sky patches in [89], including the techniques described in [70, 73, 71] and their own probabilistic model [90] which achieved the best results.

Another very recent benchmark evaluates the performance of multiple methods on ASIs [91]. The utilized dataset focuses especially on covering a high diversity of atmospheric conditions and includes 829 manually segmented ASIs in total. Multiple current state-of-the-art techniques are evaluated on this dataset (multicolor criterion [72], CSL [80], Hybrid Thresholding Algorithm (HYTA) [71], region-growing [92] and FCN [43]). Although the FCN achieved the best results, it has to be considered separately, as a cropped rectangle from the ASI is segmented.

All aforementioned methods perform only binary segmentation, classifying a pixel either as cloud or no-cloud. However, regarding precise prediction of solar irradiance, a more fine-grained differentiation between clouds is beneficial [93]. One of the first attempts was performed by Dev et al. distinguishing between thin and thick clouds in a probabilistic framework [90]. Modeling the classes (thin/thick cloud, sky) as multivariate Gaussian distribution, the joint probability distribution of classes and image regions is learned by training with a small set of 32 labeled images. Afterwards, Dev et al. proposed a U-Net, trained on the same dataset, and compared the results to their previous approach [86]. Despite the limited amount of data, the CNN performs better, in particular for thin clouds.

Recently, another method has been published [94], addressing the problem of pixel-wise classification of cloud genera as defined by the WMO in [68]. The authors propose a supervised learning-based segmentation divided into several processing steps. First, super-pixel segmentation is applied to subdivide the image into small similar regions. For each super-pixel, feature vectors are created, based on color, internal and regional texture, as well as global relationships. Due to the nature of clouds, those feature vectors may still be very similar, which is why they apply a feature transformation based on subspace alignment and metric learning to receive more discriminative features. Classification is then performed on the transformed vectors using a SVM. For validation, they compared their approach with a pretrained FCN, beating it by almost 7% points on average. They argue that their dataset of roughly 600 images is too small for a deep learning approach to learn useful features but emphasize that there is potential when more data is available for training deep networks.

3. Datasets of All-Sky Images

Selecting suitable data for training deep learning models is essential to achieve good generalization. This chapter briefly presents the framework conditions for ground-based cloud observation. The main part deals with the new dataset that was created for supervised classification and segmentation. The process of annotating ASI datasets and a comparison with other datasets from the literature is presented. In addition, the selection of ASIs for self-supervised representation learning is described. Apart from a high variety of atmospheric conditions, it is also important to have a balanced distribution.

3.1. Utilized hardware and observation site

All image data for the new dataset was collected from the Plataforma Solar de Almeria (PSA) in the desert of Tabernas (Spain) which is operated by CIEMAT[1]. The utilized All-Sky Imager is based on an off-the-shelf surveillance camera from the manufacturer Mobotix (model Q25). Figure 3.1 shows one of the installed devices on the PSA. Its maximal resolution is limited to 6 MP. However, for compatibility with the previous Q24 model, the images are cropped and resized to a resolution of 2048 x 1536 pixels (3 MP). The exposure time is set to 160 μs as previous studies showed that current state-of-the-art methods, like the CSL, work best with a fixed exposure time [81]. A Q25 model from Mobotix costs around 800 EUR in 2020. Images are obtained from a single device which is part of a meteorological station at the CSP test stand "Kontas". An image is taken every 30s from sunrise to sunset, leading to approximately 1800 images per day in summer.

3.2. Creation of an annotated All-Sky Image dataset

In the following section, the selection and annotation process for the new dataset is presented. It starts with a discussion about how the ASIs are classified, providing definitions of the utilized labels. Next, the manual segmentation and labeling process is briefly introduced. Finally, the selection of images and the variety of atmospheric conditions are discussed.

[1]Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas: A spanish research institute with focus on energy and environmental issues.

Figure 3.1.: Picture of a Mobotix camera for taking ASIs

3.2.1. Selection of images and ground truth creation

For the creation of this dataset, images were annotated on pixel- and image-level to train models for classification and segmentation. A particular feature of the classification approach is to train a model to predict multiple labels for a given sample image instead of a single class. The terminological difference between label and class is that a sample can be assigned either a unique class or multiple labels that are not necessarily mutually exclusive. In preceding studies [89, 91], a model's performance was analyzed by examining the results on different types of cloudiness and atmospheric conditions, for instance, by comparing the segmentation accuracy for clear sky and overcast conditions. In this work, a model shall also be capable of predicting such cloudiness characteristics. As mentioned in section 1.2, local pixel-level information is especially important for solar irradiance estimations in circumsolar regions. However, global information regarding overall cloudiness is also beneficial for nowcasting systems. For example, determining whether the sun is completely free from clouds is required to run qualification tests of solar power plants. Moreover, if a model reliably detects the degree of cloudiness, this information can be used to validate results from segmentation that should correspond to the prediction of the cloudiness label.

To distinguish between cloud types, a ternary categorization was chosen. As depicted in figure 3.2, the ten main cloud generas [68] can be split into three layers. Depending on where a cloud genus typically emerges, it is classified either as low-, middle-, or high-level cloud.

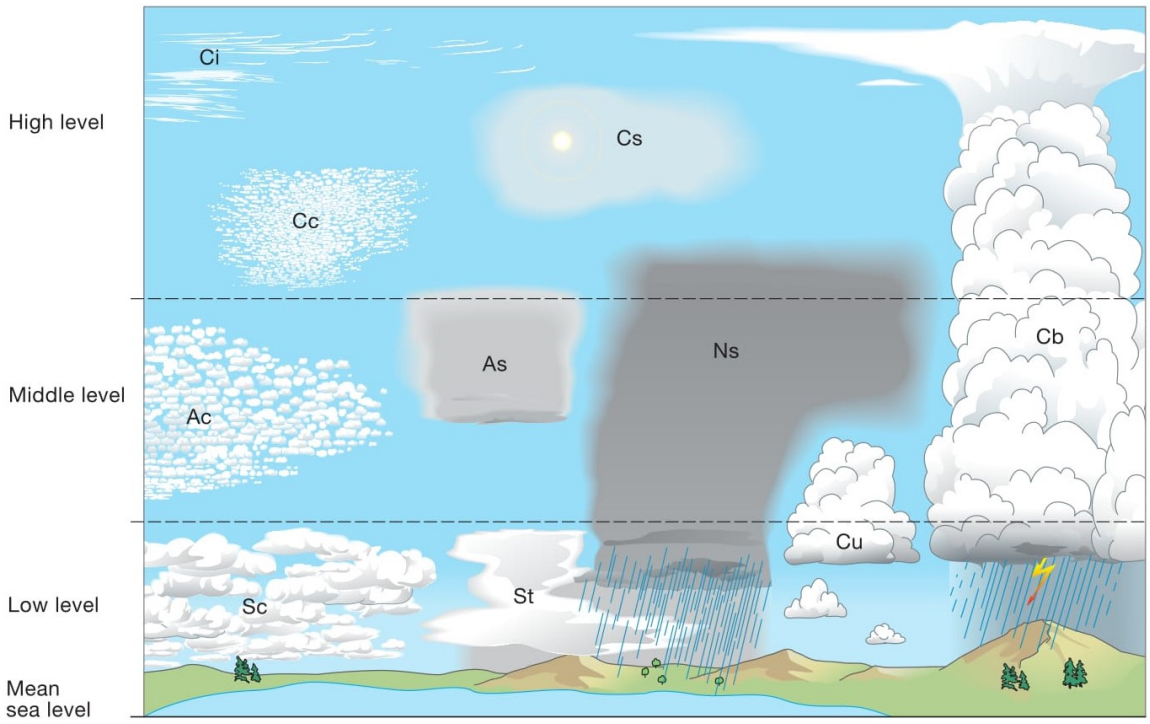
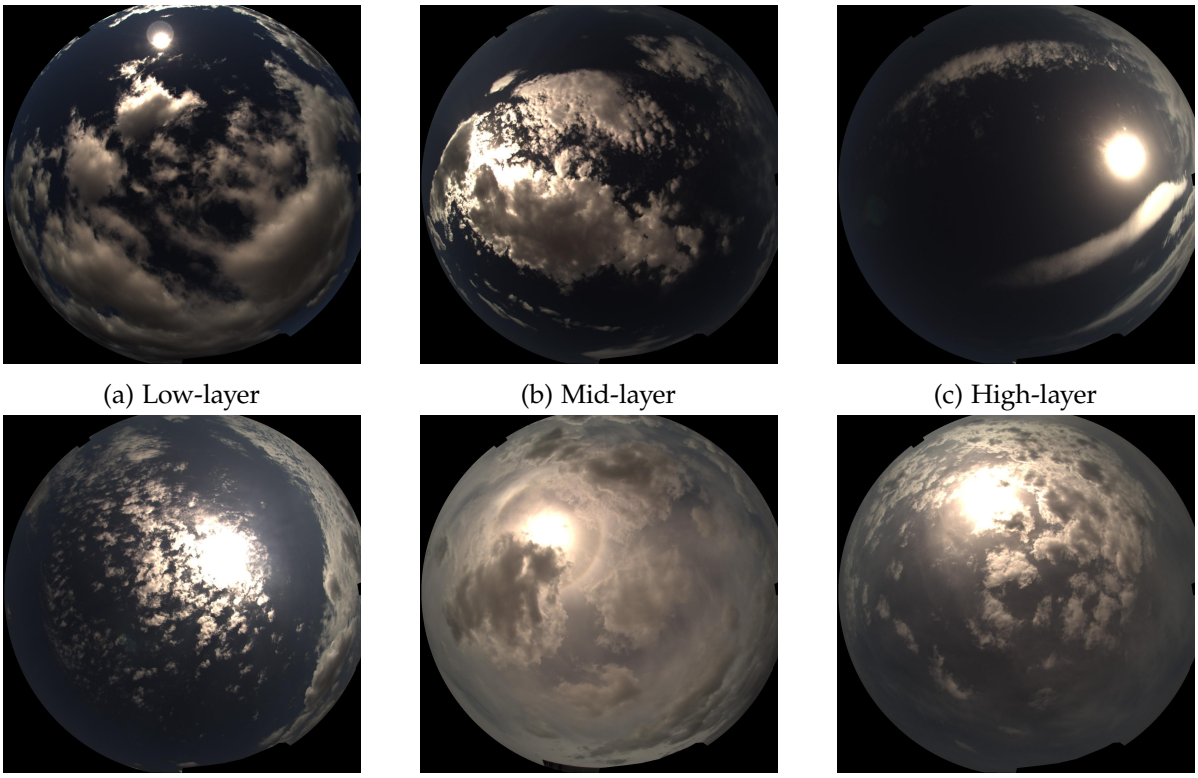


Figure 3.2.: Illustration of main cloud generas defined by the WMO [95]. The ten cloud types: Cumulus (Cu), Stratus (St), Stratocumulus (Sc), Cumulonimbus (Cb),Altocumulus (Ac), Altostratus (As), Nimbostratus (Ns), Cirrus (Ci), Cirrocumulus(CC), Cirrostratus (Cs) can be grouped by their base height.

Height, unlike altitude, indicates the vertical distance from the point of observation to the cloud. However, there is no clear border between those levels. Firstly, clouds can extend over multiple layers, as thunderclouds (Cumulonimbus) typically do. Secondly, cloud layer heights depend on current atmospheric conditions, like humidity, temperature or pressure and can vary from one day to the other. Lastly, the observation site's latitude has a significant impact on what height a cloud type occurs. Nevertheless, a subdivision of the 10 ten generas into three groups corresponding to the base height is appropriate for the approach pursued in this work. Primarily, optical characteristics of clouds within one layer are similar [93]. For example, clouds within the lowest layer are rather dense water clouds that often appear gray from beneath as a lot of solar irradiance is absorbed. On the other hand, high-layer clouds consist of ice particles only. They often appear hazy, are less densely distributed and have higher transmittance. In the mid-layer, clouds can contain water and ice particles. Hence, the transmittance is usually in between low- and high-level clouds. Another reason for this choice of categorization is the occurrence of different wind directions in the atmosphere. When detecting clouds in multiple layers, they may have distinctive directions of motion, even opposing ones. To track and predict future spatial expansion of all clouds, it is therefore

necessary to distinguish between clouds of different heights. Figure 3.3 shows exemplary clouds from single and multiple layers. In table 3.1, a summary of the presented categorization of clouds is shown with a brief description of each cloud type.



(a) Low-layer (b) Mid-layer (c) High-layer

(d) Low- & mid-layer (e) Low- & high-layer (f) Mid- & high-layer

Figure 3.3.: Examples of single and multi-layer ASI

Besides pixel-wise annotation of images indicating the cloud type, the image itself is labeled with all layers that are depicted. Hence, it can be examined if the classification model is able to detect the layers correctly regardless of their precise location. Furthermore, five other labels that can be assigned to each ASI were defined. The labels *Clear Sky*, *Overcast*, *Cloudy-SV* (sun visible), *Cloudy-SC* (sun covered) describe the form of cloudiness and are mutually exclusive. They are based on cloud coverage and the visibility of the sun. The last label is called *Sun Free* and determines if the sun disk is completely free from clouds. A summary of all labels can be found in table 3.2.

The dataset presented here is based on the data from the benchmark of binary segmentation methods in [91]. Overall, 669 images from the Kontas Q25 camera and corresponding binary masks were taken and revised. The masks were originally created with a Matlab graphical user interface (GUI) implemented using the Matlab GUIDE framework. To integrate functionality of classifying pixels into different cloud types and to ensure compatibility with future Matlab versions, a new GUI was implemented with Matlab's App Designer.

Table 3.1.: Categorization of cloud types used for semantic segmentation adapted from [95] and [96]. Height level is valid for mid-latitudes regions, like Southern Spain.

Class	Height Level	Cloud Genera	Characteristics
High-Layer	> 6 km	Cirrus	Detached, white, fibrous filaments, narrow bands
		Cirrocumulus	Thin, white, very small elements without shading
		Cirrostratus	Transparent, whitish cloud veil, fibrous or smooth
Mid-Layer	1.8 – 8 km	Altocumulus	White and/or gray, small elements with shading
		Altostratus	Grayish/Blueish layer, fibrous or uniform
Low-Layer	0 - 2.4 km	Cumulus	Sharp outlines, detached, developing vertically, dark bases
		Stratus	Fairly uniform gray base, no clear outlines
		Stratocumulus	Gray and/or whitish, visible outlines
		Cumulonimbus	Heavy, dense, large vertical extent (over all layers)
		Nimbostratus	Gray (often dark) layer, can extend to high-layer

Table 3.2.: List of image-level labels defined in the new dataset.

Label Name	Definition
Clear Sky	Less than 2% of the sky is covered with clouds and the sun disk is free.
Overcast	More than 95% of the sky is covered with clouds.
Cloudy-SV	The sky is partially clouded and the sun is visible but not necessarily free.
Cloudy-SC	The sky is partially clouded and the sun is covered such that the disk cannot be recognized.
Sun Free	No cloud is in front of the sun disk.
Low-Layer	Low-layer clouds (e.g. Cumulus) are visible.
Mid-Layer	Mid-layer clouds (e.g. Altostratus) are visible.
High-Layer	High-layer clouds (e.g. Cirrus) are visible.

The process of adding new images and creating corresponding ground truth is described briefly in the following. First, the user selects the dataset of interest which is organized via a tabular Matlab file. Next, the parameters to search for images are chosen. In addition to specifying the time frame of interest, the user can also filter for a range of TL and Solar-Zenith-Angle (SZA). If there are images matching the chosen parameters, a random sample is displayed that can be added to the dataset or discarded. Afterwards, the image is pre-segmented using a CSL to decrease the effort of manual editing. Moreover, a camera specific mask is applied which defines the region of interest within the sky image. Hence, static objects that are in the field of view of the camera can be excluded. The same way areas very close to the horizon can be ignored which are affected most by distortion effects and vignetting and thus are not clearly identifiable. Finally, the image is segmented manually using various editing tools and classified by selecting the correct checkboxes.

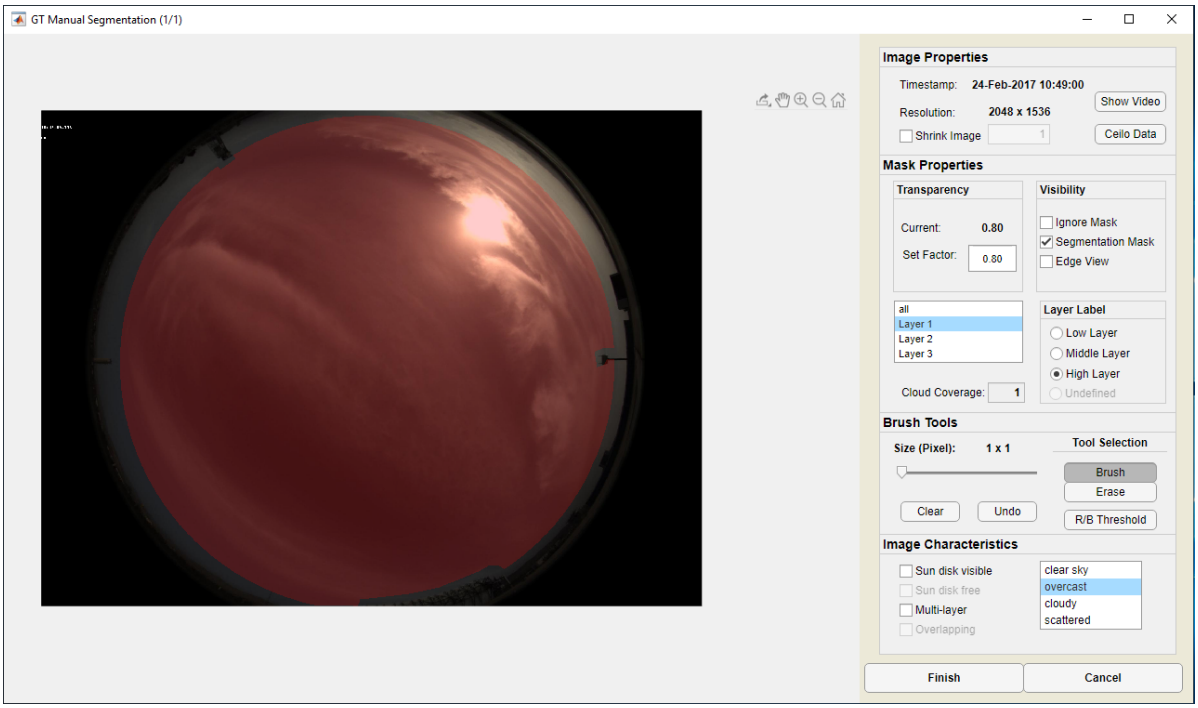
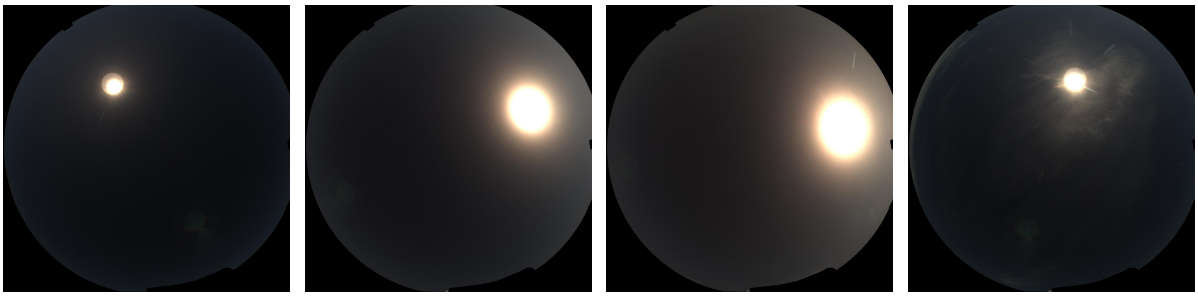


Figure 3.4.: Matlab GUI that was implemented to manually segment ASIs

A crucial difference to the original dataset is the distinction between aerosols and high-layer clouds. In most previous works, the focus was on detecting clouds that are clearly distinctive from aerosols. Hence, difficult cases showing thin ice clouds were either left out completely or these pixels were declared as clear sky. Since they still have a notable influence on solar irradiance, the dataset was extended to include these situations, too. However, aerosols and thin cloud sheets are still distinguished. As can be seen in figure 3.5, turbid atmospheric conditions affect the size of the sun disk. To check whether this mainly comes from stratus-like ice clouds, analyzing the video data proved to be helpful. If clear motion patterns can be recognized, thin high-layer clouds can be assumed. When the atmosphere is clear and the

sun disk is small, high-layer clouds may still be difficult to detect (see figure 3.5d). Cirrus clouds have more texture but are often only visible in circumsolar regions. They do not show clear contours and pixels with larger distance to the sun slowly merge with the sky.



(a) Clear sky, low TL (b) Clear sky, high TL (c) High-layer, high TL (d) High-layer, low TL

Figure 3.5.: Comparison of aerosols and high-layer clouds

In summary, existing segmentation masks were updated to include layer specification and to consider difficult high-layer pixels. Furthermore, the dataset was extended by roughly 100 images. Most of them were images with multi-layer and high-layer conditions to achieve a more balanced distribution.

To ensure that a high variety of atmospheric conditions is represented, the dataset was examined by filtering via various parameters. The following plots in figure 3.6 show the distribution corresponding to the label definitions, sun elevation, daytime, month, and TL. Generally, clear sky conditions are easy to detect and thus less images were selected. Nonetheless, different TLs are required for the model to learn to distinguish between aerosols and high-layer clouds. A large variation in sun positions was obtained by including images from an entire year and different daytimes. However, very low sun elevations ($< 8°$) were discarded. On the one hand, those images are often too dark to correctly detect clouds, as the exposure time is fixed. On the other hand, large AMs due to low sun elevation angles lead to energetically speaking less interesting conditions for solar power applications. The distribution of layers is not completely balanced, but even high-layer clouds are represented approximately every third image. Multi-layer conditions make up a quarter of the dataset. Each combination is represented similarly, apart from the scarce case with all layers visible in one image.

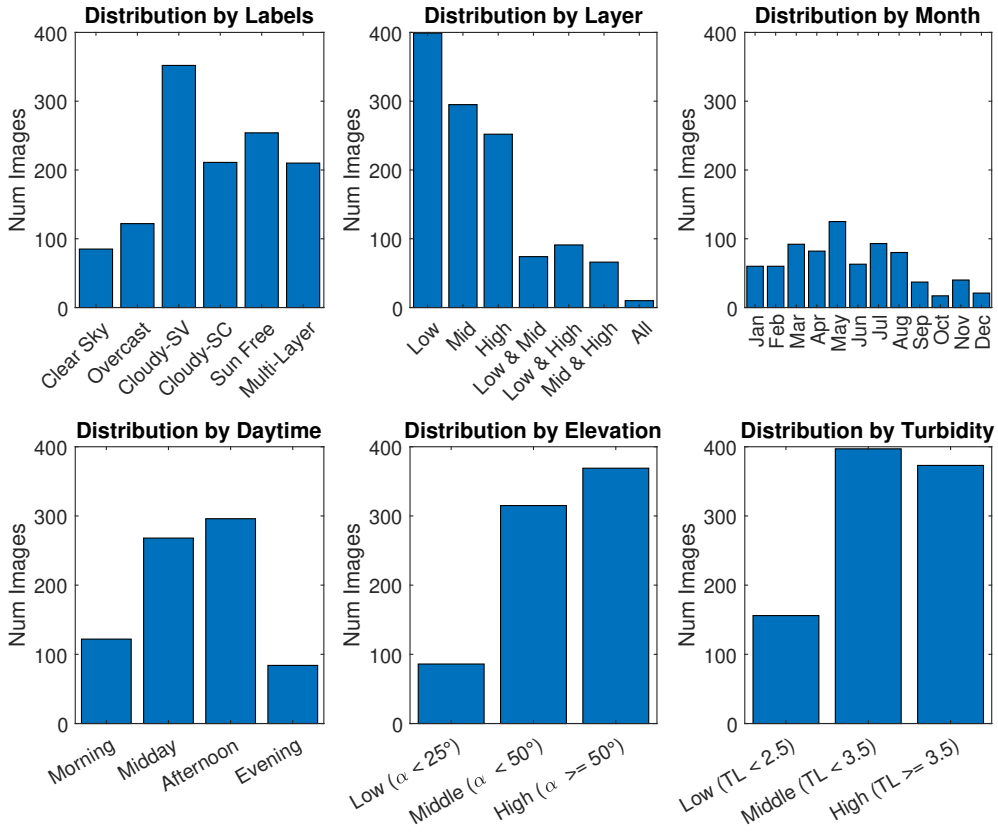


Figure 3.6.: Distribution of data according to different parameters.

3.3. Comparison to other cloud datasets

There are multiple reasons for creating a new dataset to develop deep learning models for cloud classification and segmentation. First, most datasets mentioned before do not meet the requirements for semantic cloud segmentation under multi-layer conditions, since they only include binary masks. Moreover, not all of them provide all-sky images but only sky patches. Lastly, only few datasets are publicly available. In this section, the most relevant datasets from the literature are briefly discussed and contrasted with the new dataset described in section 3.2.

In [64], approximately 1500 manually selected all-sky images were used. They were taken from a research ship on its way from Germany to Africa and thus covering different climate zones, atmospheric conditions and SZA. The ground truth includes a binary segmentation mask as well as the dominating cloud type within the image. Seven cloud types are distinguished which are based on the main generas, but merging some visually similar generas like Altostratus and Stratus. To avoid ambiguous cases, only single-layer images were selected to ensure unique classification, on average 200 per class. As the goal is to develop models that

work under all possible conditions, it is crucial to integrate complex multi-layer conditions. Hence, in this work a coarser classification was chosen, but trained and validated on a more realistic dataset.

Publicly available datasets are HYTA [71] and Singapore Whole-sky Imaging Categories/Segmentation (SWIMCAT [97], SWIMSEG [89]). All of which consist of sky patches that were extracted from ASIs. HYTA contains 32 binary segmented images of four categories: cumuliform, cirriform, stratiform and clear sky. Later a ternary version was developed in [90] where cloud pixels are categorized in terms of optical thickness: thin-/thick clouds and sky. SWIMCAT/SWIMSEG were presented in [97] to address the lack of public datasets. The former includes 784 patches in total, each annotated on image-level with one of five cloud types, namely: clear sky, patterend clouds, thick dark clouds, thick white clouds, and veil clouds. The latter contains 1013 patches with binary segmentation masks, in consideration of varying atmospheric conditions.

Another dataset of sky patches is used in [65] and adapted in [66]. It provides image-level annotations of 1231 images, originally grouped into 6 classes as proposed in [64], and later updated to match the official main cloud types. Probably the largest labeled dataset of cloud images is Cirrus Cumulus Stratus Nimbus (CCSN) [69] used for training CloudNet. It contains 2543 images of natural images of ten cloud types taken at various locations. Occasionally, some images include surrounding landscapes which is different to the other datasets that only include sky patches. Since all these datasets only capture small parts of the hemisphere, they cannot provide as much information as ASIs and are not suitable for nowcasting systems.

Recently, a very detailed dataset was presented (but not made publicly available) in [94]. It is the first approach of classifying nine main cloud types pixel-wise in ASIs. A team of experts manually created 600 ground truth masks of images obtained from two different locations in China with different air qualities.

The benchmark for binary segmentation from ASIs [91] contains 829 images and corresponding binary masks. The images were taken from two cameras and selected manually to cover a high variety of atmospheric conditions. As previously mentioned, parts of the dataset presented here were adapted from [91]. Furthermore, new images of primarily multi- and high-layer conditions were added, resulting in 770 images with respective ground truth. An overview of the described datasets can be found in table 3.3.

Table 3.3.: Comparison of datasets from the literature with the one presented here. The column "Cloud classes" indicates the total number of classes used in the ground truth (including cloudless sky). "Multi-layer" refers to the existence of multiple cloud types within a sample. "Semantic" indicates distinction of cloud types in segmentation.

Dataset	Image type	Size	Segmentation	Classification		Multi-layer
				Level	Classes	
Heinle et al.[64]	ASI	1500	binary	image	7	✗
HYTA[71, 90]	ASI	32	semantic	pixel	3	✗
SWIMCAT[97]	Sky-Patch	784	-	image	5	✗
SWIMSEG[89]	Sky-Patch	1013	binary	-	-	✗
Zhuo et al.[65, 66]	Sky-Patch	1231	-	image	9	✗
CCSN[69]	Sky-Patch	2543	-	image	10	✗
Ye et al.[94]	ASI	600	semantic	pixel	9	✓
Hasenbalg et al.[91]	ASI	829	binary	-	-	✗
Dataset presented here	ASI	770	semantic	pixel	4	✓

3.4. Dataset for self-supervised pretraining

To learn abstract representations in deep layers that generalize well, large datasets are required. Especially for segmentation, manually annotating images pixel-wise is extremely time-consuming. Assuming a person edits 5 images per hour, it would take over 20 years to obtain a dataset of 1 million images. Hence, all existing datasets of ASI for supervised learning are limited to a few hundred images. Compared to the scale of ImageNet, with 1.3 Million images [25], those are smaller by a factor of 1000 or more. Also segmentation datasets like COCO [46] contain hundreds of thousands labeled samples.

As mentioned in section 2.1, it is also questioned whether applying transfer learning from models trained on ImageNet is the best approach. If there are large amounts of raw data available, self-supervised pretraining is considered as a promising alternative. Multiple cameras are installed at PSA, taking pictures every 30s during daytime since 2015. A challenge that comes with processing big datasets is the increase in computation time. To evaluate the approach of self-supervised representation learning, images were again taken from the Kontas camera from 2017. However, it would be a waste of computational resources if all data from that year were used. At the observation site 60-70% are taken during clear sky conditions which are not very valuable for training. Moreover, clouds within images of low sun elevations early in the morning and late in the evening are often not clearly identifiable due to distortion effects and low brightness.

Therefore, the data was filtered to exclude dispensable images. To reduce the amount of uncertain situations, only images within a fixed time frame from 9am to 6pm in Central European Time were considered. While there are some images with low sun elevation in winter, the majority covers daytimes where clouds are clearly detectable. Moreover, elevation

angles lower than 6° are completely omitted as the camera automatically stops operating at night. Furthermore, the data was filtered using DNI measurements that were taken at PSA. Following the procedure described in [98], each image is assigned a DNI class from 1 to 8 corresponding to DNI measurements of the preceding 15 minutes. Clear sky conditions with very low to medium variability in DNI are described by classes 1 to 3. Class 4 has still a high average DNI but with stronger temporal variability. For classes 5 and 6 the average DNI is significantly lower while class 7 represents almost opaque overcast situations with only few ramps. Class 8 describes fully overcast situations with constantly low DNI.

In figure 3.7, the distribution of images before and after DNI filtering is depicted. It shows the relative distribution of images grouped by DNI classes in the dataset. Mainly images from DNI class 1 and 2 were filtered out such that a balanced distribution is attained. In the end, the dataset for self-supervision includes 286477 images which is roughly 55% of all images taken in 2017.

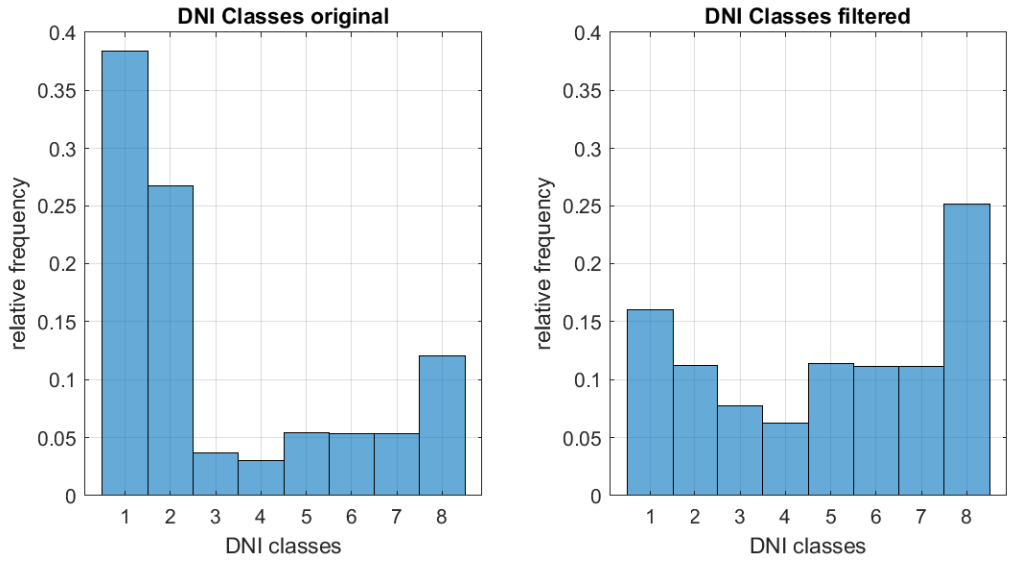


Figure 3.7.: Filtering ASI by DNI classes to reduce computational effort in training.

4. Implementation and training of deep learning models

In this chapter, the models for supervised and self-supervised learning are presented. The first three sections deal with supervised multi-label classification and semantic segmentation. Decisions regarding network architecture are discussed and implementation frameworks are mentioned. Also the applied methods for training are described briefly. In the second part, the pretext tasks that were applied for self-supervised learning are presented.

4.1. Multi-label classification model

Usually when classifying a sample, there is only one correct class that it is assigned to. In the presented dataset, there are multiple labels that can be correct simultaneously. This is because the labels are defined to describe different characteristics of ASIs (see table 3.2). Although some of the labels tend to occur together, they are mainly independent of each other. For example, clear sky can be true, although distant high-layer clouds are visible. On the other hand, high-layer clouds could cover the entire ASI indicating overcast conditions.

In standard classification with more than two classes, a model estimates the probability for each class in relation to all other classes. A target vector $\hat{\boldsymbol{y}}$ of size C (number of classes) is usually represented in one-hot encoded format. Thus, all entries are zero except the one at index i_c representing the target class. Typically, the softmax function σ is applied to the score $\boldsymbol{z}$ from the network's last layer to predict a unique class.

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \quad j = 1...C$$

It assigns each label a normalized probability value and ensures that the sum $\boldsymbol{y} = \sigma(\boldsymbol{z})$ equals to one. The predicted class is then determined by applying argmax.

For multi-label classification, the model has to be capable of predicting multiple labels for a given input, thus softmax is no suitable choice. A straightforward approach is to decompose the task into N_c binary classification problems, also known as binary relevance learning [99]. Each label is predicted independently by applying the logistic function to each score value z_j.

$$\tilde{\sigma}(\boldsymbol{z})_j = \frac{1}{1 + e^{z_j}}$$

Here, the output is also between 0 and 1. However, it is not normalized with respect to the other label predictions but it only depends on the score value for the respective label. Finally, a predefined threshold determines whether a label was detected or not.

As architecture ResNet34 [42] was chosen. It is based on the residual learning framework mentioned in section 2.1 which introduced skip connections and made training of very deep networks feasible. The core of any ResNet is a concatenation of multiple residual blocks. Each contains convolutional layers that are followed by BN and a single ReLU per block. The skip connection is in between the residual blocks, hence skipping two convolutions. A detailed list of components and operations can be found at table 4.1. A diagram visualizing the model is illustrated in figure 4.1. Although there are deeper versions of ResNet architectures which performed better on benchmarks like ImageNet, the 34-version is a suitable choice. To learn very deep features, extensive training is required [32]. Consequently, also lots of data is necessary. However, our dataset is limited to 770 images, which is significantly less than 1.3 million images in ImageNet. Regarding transfer learning from ImageNet pretraining, weights from deeper layers do not add much value, as these layers look for specific features of ImageNet classes that are not relevant in ASIs. Furthermore, deeper architectures need more computational resources, especially in terms of GPU memory.

Table 4.1.: Structure and components of a ResNet34 applied to ASIs of size 512x512. The network consists of 33 convolutional and one fully connected layer. Batch Normalization (BN) is applied after each convolution. The first convolution of each ConvX_1 has stride 2.

Group	Output size		Components	Filter Size	Stride
Conv1	256x256x64	1x	Conv→BN→ReLU	7x7	2
MaxPool	128x128x64	1x	Max Pooling	3x3	2
Conv2_x	128x128x64	3x	Conv→BN→ReLU	3x3	1 (2)
			Conv→BN	3x3	1
Conv3_x	64x64x128	4x	Conv→BN→ReLU	3x3	1 (2)
			Conv→BN→ReLU	3x3	1
Conv4_x	32x32x256	6x	Conv→BN→ReLU	3x3	1 (2)
			Conv→BN	3x3	1
Conv5_x	16x16x512	3x	Conv→BN→ReLU	3x3	1 (2)
			Conv→BN	3x3	1
AvgPool	1x1x512	1x	Avg Pooling	16x16	
FC	8	1x	Fully Connected		

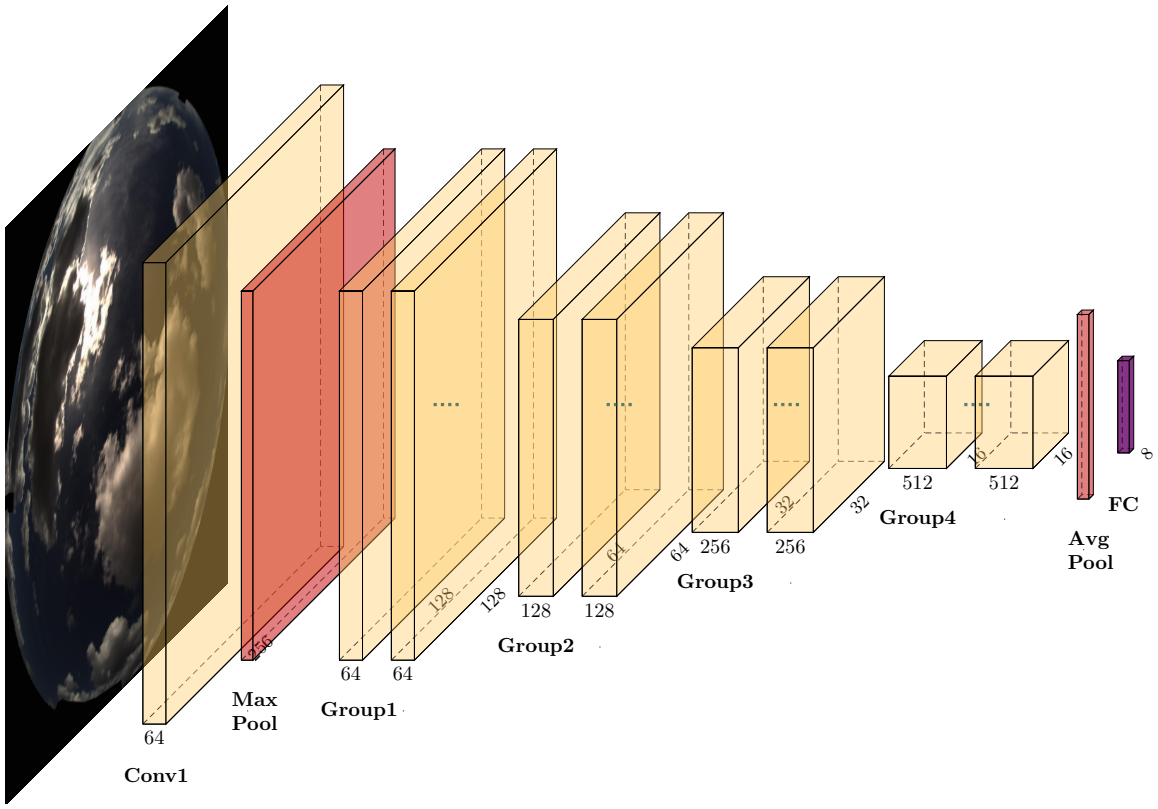


Figure 4.1.: ResNet34 Architecture for Multi-Label Classification. Each group contains a concatenation of 3 to 6 residual blocks.

4.2. Semantic segmentation model

To obtain pixel-wise predictions, a model based on the U-Net architecture [44] was implemented. Like other FCN, it uses deconvolutions to upsample dense representations from the downsampling path. It is called downsampling because the network reduces feature map size by applying pooling operations or convolutions with strides greater than 1. It can also be regarded as encoder whereas the upsampling path decodes the extracted features to associate them spatially within the image. The encoder part can consist of any downsampling CNN architecture. Here, again a ResNet34 was utilized. Aside from the average pooling and the fully-connected layer, the ResNet34 is the same as for multi-label classification. Deconvolutions are typically composed of two steps: (1) expanding the input feature maps, (2) applying one or multiple convolutions followed by non-linear mappings. The expansion operation simply creates new pixels in between the existing ones. Crucial is how these gaps are filled. The straightforward approach, as in the original FCN [43], is bilinear interpolation. However, other strategies turned out to work better. In this work, the so-called pixel shuffle method [100] with ICNR (Initialization to Convolution NN Resize) [101] is applied for upsampling.

Each deconvolution thereby doubles the resolution, matching the sizes of intermediate feature maps from the encoder.

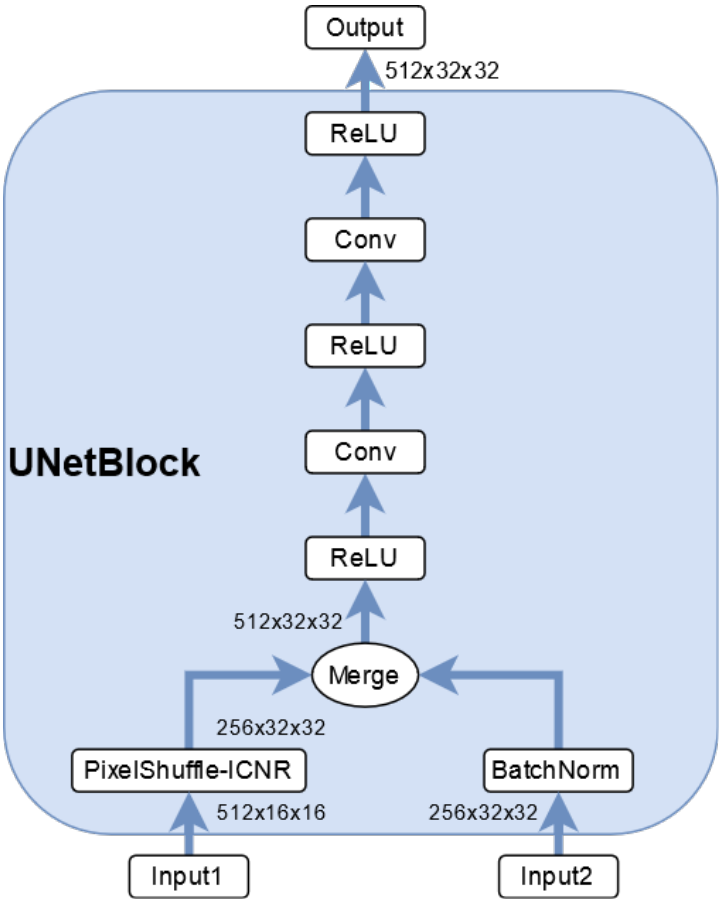


Figure 4.2.: Diagram of deconvolution block within the U-Net architecture. Sizes are exemplary to show doubling of resolution. Input1 refers to the output from the preceding layer whereas Input2 indicates a skip connection from the corresponding layer in the encoder part.

The particular characteristic in U-Nets is that there are skip connections between encoder and decoder. After every deconvolution, the output of an encoder layer with corresponding resolution is merged. The resulting tensor serves as input for further convolutions to learn dependencies. This procedure of stacking previous feature maps on deconvolution results, followed by convolutions constitutes a U-Net block. A diagram of such a U-Net block is depicted in figure 4.2. In total, there are four such blocks upsampling the feature maps to 256x256. Another deconvolution leads to the original resolution (512x512) and 96 filters. Merging this tensor with the original input image and applying a final convolution results in the final output tensor of 5x512x512. The values of each channel $c = 0...4$ represents the scores at the pixel position (x, y) for the respective class. By Applying softmax and argmax pixel-wise over all channels (as described in section 4.1), a unique class is predicted.

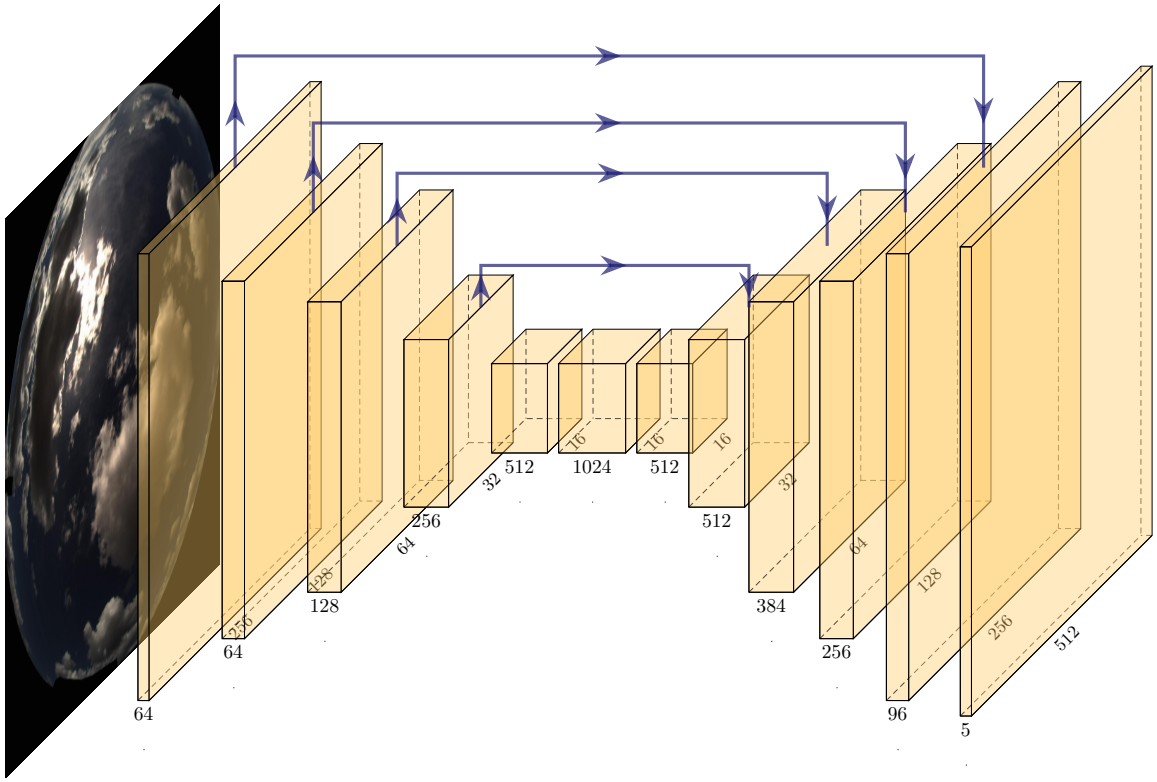


Figure 4.3.: Simplified graph of U-Net architecture for segmentation with ResNet34 backbone. Pooling layers are omitted and groups of residual blocks (left) are combined. On the right, the cuboids represent the U-Net blocks and the final layer (rightmost).

A dummy class is added to represent surrounding border area that results from the fish-eye image. Overall, the classes *border*, *sky*, *low-layer*, *mid-layer*, *high-layer* correspond the the channels 0 to 4 respectively.

To compare the deep learning approach with state-of-the-art segmentation techniques, another model was trained to solely perform binary segmentation using the classes *cloud* and *sky* (and *border*). In chapter 5, it is then evaluated if retraining a binary model is needed or if postprocessing the semantic segmentation achieves equal performance.

4.3. Supervised training

In this section, all relevant information regarding the training procedure is described. In particular, data augmentation, initialization and the selection of hyperparameters are discussed. For implementation, I used a layered API built on top of Pytorch called fastai [102]. It is written in Python 3 and provides high-level components to implement state-of-the-art models, as well as low-level features for development. The current version that was utilized in this

work is fastai v1.0. Training was performed on a single *Nvidia GeForce RTX 2080 Super GPU* with 8GB RAM.

4.3.1. Data augmentation and normalization

Data augmentation is particularly useful for small datasets, as it supports the model's generalization ability. By transforming the input data randomly when a batch is loaded, the risk of overfitting is reduced. Nevertheless, transformations have to be reasonably chosen. Important features for recognition should not get lost and they should preserve the labels. For instance, if an ASI would be cropped randomly, removing the sun from the image, the labels would not fit anymore. Before discussing the applied data augmentations techniques, another essential transformation should be mentioned.

In general, image size has to be consistent for a CNN. Hence, it is often necessary to resize the images such that the input is always equal and suitable for the task. Commonly, classification models are trained with resolutions of 256x256 or lower. For instance, the CIFAR-10 dataset [103] contains images of size 32x32 which is still sufficient to recognize depicted objects but requiring significantly less computational effort to process. Many popular architectures benchmarked on ImageNet, like AlexNet [30], use an input size of 224x224. This was also adapted on some models for cloud recognition. In [69] or [66], the input size was set to 224x224, too. However, the utilized datasets contain only sky-patches. In ASIs, a much larger region of the sky is depicted. Moreover, with increasing distance from the zenith, the observed clouds are much further away, especially in the case of high-layer clouds, and have lower resolution due to small viewing angles. To avoid to loose relevant information in these regions, larger input sizes are reasonable. Eventually, it is a trade-off between computational costs and benefits of more detailed images. For the models in this work, the input size was set to 512x512 which corresponds to other recent works that used ASIs as input for CNNs [87, 104].

The same set of augmentation techniques was used for classification and segmentation, focusing mostly on affine transformations. As All-Sky Imagers take pictures from the entire hemisphere, there is no top or bottom. Hence, rotations and reflections are very suitable to artificially inflate the dataset. Here, rotations of 90° in both directions, as well as vertical and horizontal flips were applied. Brightness was slightly adapted by increasing or decreasing it up to 10%. All transformations were applied independently with the probability $p = 0.75$ for every batch during training.

Apart from data augmentation, normalizing the input is also an important preprocessing step to efficiently train a network and facilitate convergence [37]. By default, the input range for each color channel in Pytorch is scaled to $[0, 1]$. Depending on initialization, the input is possibly further normalized. When the model is initialized with pretrained weights, the input is normalized with the mean and standard deviation of the pretraining dataset [36]. Precisely, the mean is subtracted channel-wise from the input and then divided by the standard deviation.

4.3.2. Weight initialization

Weight initialization in deep learning is one of the most relevant aspects for successful training. Studies like [37, 38, 39, 40] analyzed how to effectively initialize weights to train a model from scratch. In this case, as the training set is relatively small, the focus is on transferring pretrained weights and only fine-tune on the annotated data. These weights can be obtained from publicly available models or from self-supervised ones. In the first approach, a ResNet34 pretrained on ImageNet is used which is available in Pytorch. Additionally, weights are also transferred from two pretext tasks: Inpainting-Superresolution (IP-SR) and DeepCluster (DC). However, for both models, classification and segmentation, not all layers can be initialized with pretrained weights. Regarding classification, the last fully-connected layer is always task specific. Here, it maps the feature map from the average pooling layer to the eight labels and is randomly initialized. For the U-Net model, the entire decoder part starts with random weights. As the purpose of pretraining is to obtain better representations, only the weights from the encoder part are transferred. Besides, pretrained models like the ones trained on ImageNet do not include deconvolutions. The common technique for weight initialization using ReLUs as non-linearities is Kaiming-initialization [40]. Basically, the weights are initialized by drawing from a zero-based Gaussian distribution $\mathcal{N}(0, \sigma^2)$ whereas the standard deviation σ is determined for each layer l separately. More precisely, the number of connections of the input $n = k^2 c$ is decisive, with k indicating the filter size and c specifying the number of filters. The resulting standard deviation is then computed by:

$$\sigma = \sqrt{\frac{2}{n_l}}$$

To validate and compare the transfer learning approaches, the models for classification and segmentation are also evaluated with complete random initialization in sections 5.2 and 5.3.

4.3.3. Loss functions, optimizer and hyperparameter selection

As all labels are considered separately in the multi-label classification task, the error corresponding to a specific label has to be computed independently of all others. Binary-cross entropy is a common and suitable choice for this:

$$\mathcal{L}_{n,c}(y_{n,c}, \hat{y}_{n,c}) = \hat{y}_{n,c} \cdot \log \sigma(y_{n,c}) + (1 - \hat{y}_{n,c}) \cdot \log(1 - \sigma(y_{n,c}))$$

where $y_{n,c}$ refers to the prediction of a sample n with respect to label c and $\hat{y}_{n,c}$ to the corresponding target. $\sigma(x)$ indicates the logistic function.

For segmentation, a pixel-wise categorical cross-entropy is applied to compute the loss:

$$\mathcal{L}_{n,i,j}(y_{n,i,j}, \hat{c}) = -y_{n,i,j,\hat{c}} + \log\left(\exp\left(\sum_{k=0}^{C} y_{n,i,j,k}\right)\right)$$

$\hat{c}$ is the index in the range $[0, C]$, corresponding to the target class of sample n at pixel position (i, j). If the model is trained for binary segmentation $C = 2$, otherwise, when distinguishing

between cloud layers, $C = 4$. For every mini-batch, the loss function is computed, averaged and the error is backpropagated to compute the gradients. Updating the learnable parameters within the networks for both tasks is achieved using the stochastic optimization method Adam (Adaptive moment estimation) [105]. The corresponding hyperparameters are set to the default values as proposed in [105] ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10e^{-8}$).

For optimization, a very important hyperparameter is the learning rate (LR). It is essential to achieve fast convergence. Recent studies propose a systematical approach to identify suitable values. In [106], it is suggested to select the LR by training the network for one epoch while incrementally increasing the LR for each mini-batch. By examining the loss curve, divergence can be recognized easily. Figure 4.4 shows an exemplary graph of the LR finder method. A suitable LR can then be chosen by selecting a value left from the minimum. As analyzed in [106], an order of magnitude ten smaller is a sensible choice.

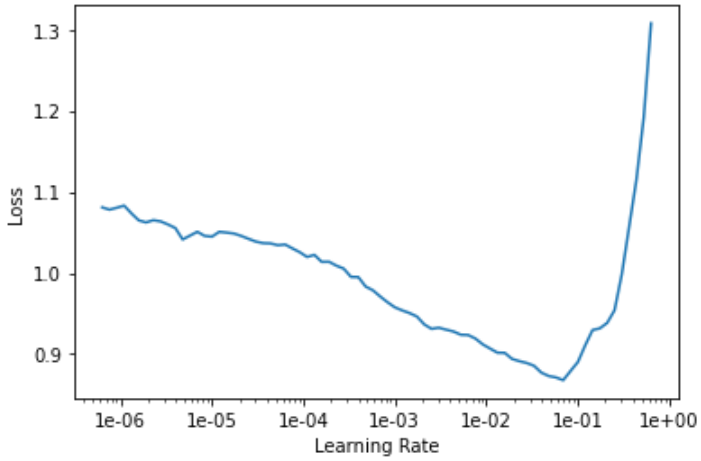


Figure 4.4.: Loss with respect to LR. By incrementally increasing the LR for each mini-batch, the loss first drops until a proper LR is reached and then rises fast as training starts to diverge.

Another concept from [106] dealing with LRs is the one-cycle policy. Instead of training with the same LR over the entire training process, it is proposed to start with a smaller value, increase it until the maximum is reached and decrease it again. At the very end, for the last few epochs, the model should then be trained further with an even smaller LR. The idea is to start and stop slowly. Due to the increasing LR, the model does not get stuck into early and steep local minima but searches for smoother regions that minimize the loss. With a small LR in the end however, the model can then reach a steeper minimum from the smoother surrounding. Thereby, the one-cycle policy acts also as a regularization technique [107]. A typical progression of the LR in this method can be seen in figure 4.5. Lastly, weight decay is applied as further regularization technique as described in [108].

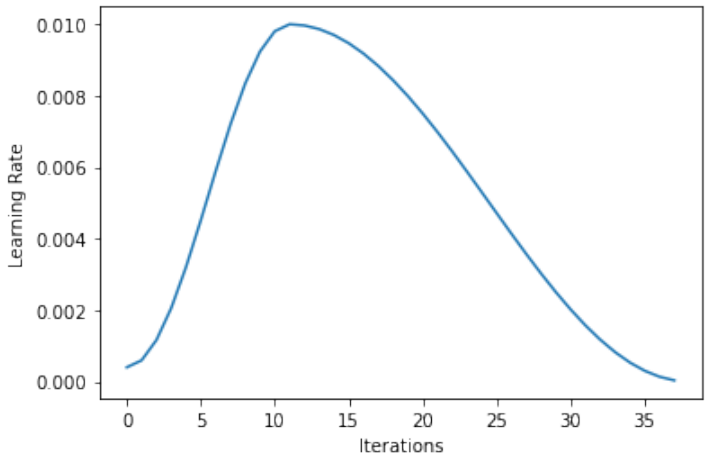


Figure 4.5.: LR with respect to training iterations. The LR starts small, increases until the maximum is reached and then decreases again. The last epochs are trained with even lower LR.

4.3.4. Splitting data into training and validation set

Dividing available data into a subset for training and validation is an essential procedure to observe the model's generalizability. Some data is excluded entirely from the training process by using a fixed validation set of 20% and measuring the performance on unseen samples. To compare the models with different initializations, it is further required to always use the same samples for training and validation. Otherwise, a validation set may contain ASIs with easier or more difficult conditions leading to different performance results. This is in particular due to the relatively small size of the validation set containing 154 images in total. In the extreme scenario, if all clear sky images would fall into the validation set, the model would most likely not be able to detect clear sky conditions. Therefore, an unbalanced distribution of ASIs should be avoided. A straight forward approach to prevent this is by checking if a random selection is balanced. If not, new samples are selected randomly until the criteria for a balanced distribution are met. By setting up the requirement that at least 19% of each label should be present for validation, a list of samples assuring balanced distribution is generated.

4.4. Self-supervised pretext tasks

Exploiting the availability of large amounts of raw data is a key principle in self-supervised learning. As described in section 3.4, a dataset of over 280,000 images was created to use for pretraining which is roughly 300 times more than the annotated data. This can help to learn more distinctive features needed to detect the different cloud types, as variety in the unlabeled dataset is much higher.

Selecting a suitable pretext task is the most crucial factor to learn useful representations for the downstream task, i.e the classification and segmentation tasks. The goal is to train a

network that solves a problem by extracting the relevant features of the three cloud types. Hence, the model should acquire an understanding of the data and be able to recognize different cloud conditions without naming them. To achieve this, the task must not be too difficult and should be solvable by humans. On the other hand, it is important to prevent short-cut solutions that allow the model to solve the problem without detecting those features. For instance, due to the distortion from the fish-eye lens, solving a jigsaw puzzle [61] could possibly be achieved by learning the distortion itself independently of the apparent clouds.

Two different pretext tasks were implemented to learn visual representations with self-supervision. The first is based on a generation-based approach, combining two methods from the literature: inpainting [57] and superresolution [59]. For the second task, the DeepCluster technique [63] was adapted for ASIs.

4.4.1. Inpainting and superresolution

In the first approach, the model aims to solve two problems. Similar to [57], multiple randomly black boxes of fixed size were inserted into the images. The idea is that the model has to detect the surrounding conditions in order to generate patches that fit into the context. For example, if the black box is within a sheet of thin high-layer clouds, it has to be filled differently than when the surroundings are part of a dense low-layer cloud. Moreover, when a black box covers the sun, the model could learn to detect the sun by inspecting circumsolar regions. Generally, the effects caused by the sun in circumsolar regions under certain turbidities could thus be learned.

As second task, the model needs to increase the resolution of the input image. Therefore, the original resolution was first reduced by half, resulting in an image size of 256x256. To restore the original size of 512x512, the image was then scaled up again using bilinear interpolation. The purpose is in particular to learn structural and textural characteristics of different cloud types. For instance, that Cumulus-shaped clouds have sharper borders than hazy Cirrus clouds. Although GANs proved to work well for this purpose (e.g. [58]), I did not use them in this approach. A common problem with GANs is that they are often not easy to train, especially in the beginning, and training takes longer as two networks have to be trained simultaneously. Another drawback with training two networks at the same time is the increased amount of Graphical Processing Unit (GPU) memory that is needed. It turned out that this was also a limiting factor for me when using input sizes of 512x512 as even a batch size of 2 would have required too much GPU memory. The technique described in [59] represents a promising alternative that requires less computation time and memory. The principle of perceptual losses is to add an extra term to the loss that comes from comparing intermediate activations from a trained network when feeding the original and the generated image, see figure 4.6.

Again a ResNet34 serves as architecture, nested in a U-Net to generate images as output. In this first study, 4 quadratic boxes were randomly distributed in all images, each box having an edge length of 80 pixels. This size guarantees to possibly cover the sun and smaller clouds, without covering larger parts of the image, potentially concealing major clouds of other layers.

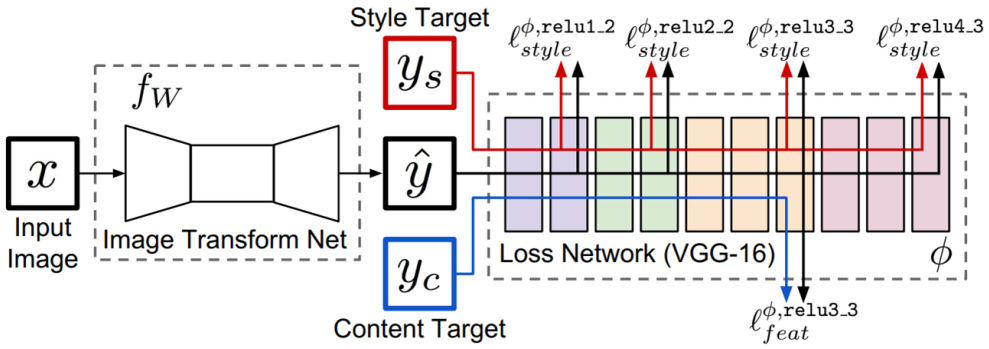
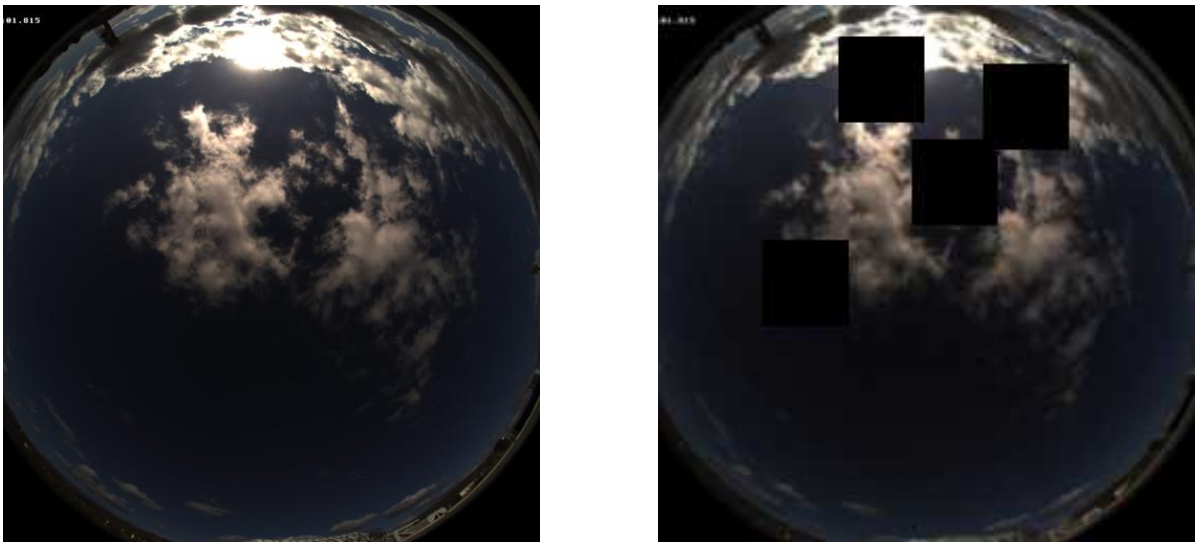


Figure 4.6.: Illustrating the principle of perceptual losses obtained from [59]. On the left, the generating network is depicted yielding the prediction y. The right shows a pretrained and fixed loss network. Here, only feature losses l^{ϕ}_{feat} are used which measure differences between activations in multiple layers for a target y_c and a generated image y.



(a) Original image

(b) Corrupted image

Figure 4.7.: Example of input (right) and ground truth (left) image for inpainting/superresolution pretext task.

Choosing four boxes leads to maximum 10% of erased pixels such that the remaining ones suffice for human observers to recognize surrounding characteristics. An example showing the original and the corrupted image can be found in figure 4.7. The loss is composed of the previously mentioned perceptual loss and a pixel-wise Mean Absolute Error (MAE). As loss network the proposed VGG-16 [36] pretrained on ImageNet was utilized. Perceptual losses are computed from four hidden layers which directly precede a Max-Pooling layer. The model's weights are again initialized with Kaiming initialization and the learning rate was determined with the LR finder. Also, one-cycle policy is applied and Adam is used for optimization.

4.4.2. DeepCluster

The second pretext task refers to the DeepCluster method described in [63]. A recent survey on self-supervised learning [52] compared various approaches, among others also DeepCluster, which achieved the best results for classification and segmentation downstream tasks by a large margin.

Training a DeepCluster network consists of two alternating steps. The first is to cluster the output features of a CNN with a standard k-means clustering algorithm. Using those assignments, pseudo-labels are generated for the images. The second step is basically solving a standard classification problem. The network is trained with the pseudo-labels using backpropagation and stochastic optimization techniques. This procedure is repeated for a specified number of iterations, switching between clustering and classification every epoch, see also figure 4.8.

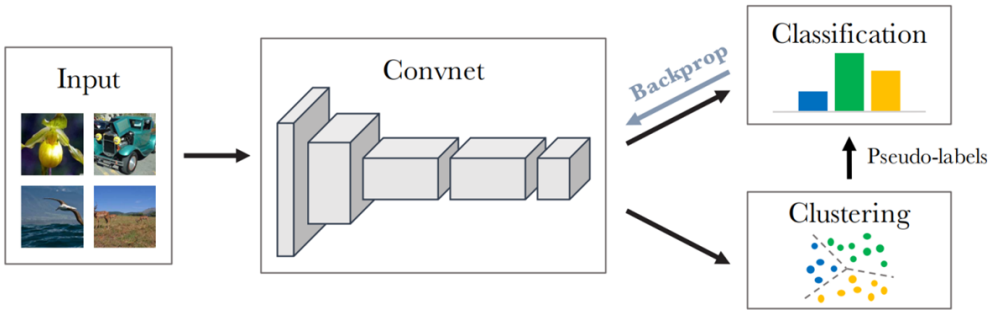


Figure 4.8.: Principle of the DeepCluster method (graphic adopted from [63]). Each epoch, the model switches between classification and clustering.

In [63], an AlexNet and a VGG-16 were applied as CNN architectures. To transfer the weights for the downstream tasks, in this work a ResNet34 was utilized. Furthermore, the CNN is divided into three parts. A *features* part, containing all the convolutional layers of the ResNet34, a *classifier*, consisting of two FC layers with dropout in between and ReLU activations, and a *top-layer*, representing another single FC layer. The latter is trained from scratch for each epoch, as preceding clustering leads to new distributions of the pseudo-labels.

Unlike in the original DeepCluster, no Sobel filters were applied to remove color. In [63] and other works [60, 61], it is argued that self-supervised learning can have difficulties when being trained directly on color images. For example, when a model was trained with the DeepCluster method on raw color images, the first layer contained a lot of filters capturing less relevant color information [63]. For cloud detection however, color information, like RB-ratio is essential (see section 2.2.2), so removing this information would be inconvenient.

Before clustering the features from the second FC layer in the classifier part, they are preprocessed with PCA, whitening and L2 (least-squares) normalization. In particular, PCA is important to reduce the dimension of the flattened feature vector to 256, allowing faster clustering. Nonetheless, as the entire dataset needs to be clustered, a GPU-based approach is necessary to allow parallel computing. As in [63], facebook's library *faiss* [109] based on the corresponding paper [110] was used in this work, too.

An important hyperparameter for DeepCluster is the number of clusters k to group the features. Analyzing DeepCluster's performance for different ks, it was concluded that a higher value compared to the number of classes within the dataset is beneficial. As training was performed on unlabeled images from ImageNet, the best performance was obtained with $k = 10000$ which is 10 times more than the number of classes. For the unlabeled ASI dataset, there is no fixed number of classes. A suitable k could be derived from the classification of cloud types used in this work, i.e low-layer, mid-layer, and high-layer, alternatively from the ten main cloud generas. However, it is more likely that ASIs are clustered especially in terms of overall image characteristics. For example by grouping overcast and clear sky images, or any degree of cloud coverage. Also, different turbidities are likely to be considered for clustering. Hence, to force the model to differentiate between cloud types, a higher value for k seems reasonable. In chapter 5, the performance of DeepCluster with $k \in \{30, 100, 1000\}$ is evaluated. Other hyperparameters were mostly adapted from the original paper [63].

4.4.3. Normalizing the data

As described in section 4.3, normalization of the input is a standard preprocessing step, that scales the RGB data to the range [0,1], substracts the dataset's mean μ and divides by the standard deviation σ. Whereas μ and σ are computed for each color channel respectively. In ASIs however, a significant fraction of the image only consists of black pixels that would distort the normalization factors. Therefore, they were explicitly excluded from computing μ and σ, resulting in the following RGB values (also scaled between 0 and 1):

$$\mu = [0.173905, 0.169622, 0.171537]$$
$$\sigma = [0.137648, 0.129713, 0.117508]$$

5. Experimental Results

Multi-label classification and semantic segmentation of ASIs using deep learning techniques are the two major goals of this thesis. In the preceding chapters, utilized architectures and techniques were presented to train the CNN models, as well as the new dataset. To address the issue of limited data in supervised learning, two approaches for learning visual representations with self-supervised pretraining were described.

In this chapter, training results for the pretext tasks and their effectiveness for applying the trained weights for the classification and segmentation tasks are discussed. The self-supervised models are compared with a random initialized model and another pretrained on ImageNet. This is achieved by fine-tuning all models with the annotated dataset for the same amount of epochs. Figure 5.1 shows the setup of the presented approach.

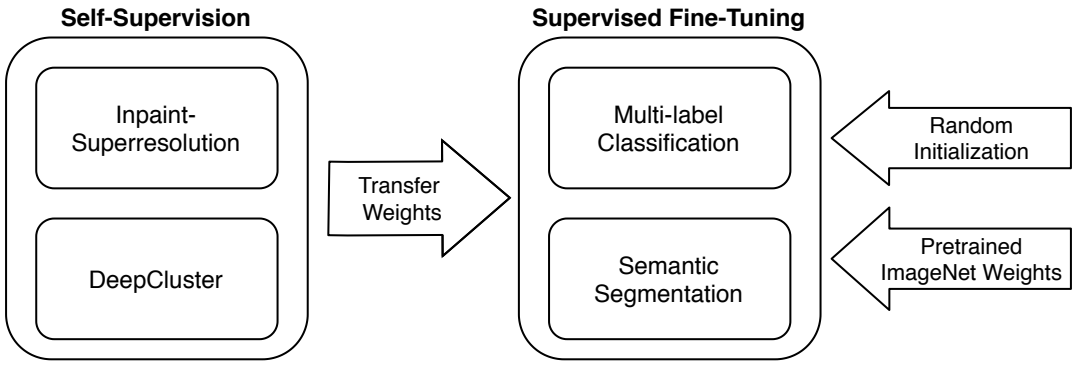


Figure 5.1.: Setup of our supervised multi-label classification and semantic segmentation tasks.

For multi-label classification, *accuracy* and *F1-score* are used to evaluate performance. The metrics for semantic segmentation are *precision*, *recall*, *Intersection over Union (IoU)* [45] and *pixel-accuracy*. To interpret strengths and weaknesses of the model, some of the results are visualized. Furthermore, it is determined whether a model trained for binary segmentation of ASIs is more effective than preprocessing the predictions of the 4-class segmentation model. Finally, as validation against a state-of-the-art segmentation approach, the performance of binary segmentation is compared with a CSL [81], previously described in section 2.2.2.

5.1. Self-supervised pretext tasks

At first, the self-supervised pretext tasks are discussed. The training settings are briefly revised and the results are examined. In particular, it is analyzed how well the models perform on

solving the specified problem by visually inspecting some of the results. The suitability of self-supervision for the downstream tasks (classification/segmentation) is discussed later, in sections 5.2 and 5.3.

5.1.1. Inpainting-Superresolution

As described in 4.4.1, a U-Net model with a ResNet34 backbone was used to predict missing parts in ASIs as well as to increase the resolution. The same data augmentation was applied as for the supervised training and normalization using the dataset's specific mean and standard deviation (see section 4.3 and 4.4.3).

The LR finder was used to determine a suitable value and the one-cycle policy was applied for training [106]. The validation set was determined by randomly splitting 20% from the total dataset, resulting in 57295 images. No cross-validation was applied and the images from the validation set are never used for model optimization.

In total, the model was trained for 10 epochs with batch size 2 and an average computation time of 11 hours per epoch. All training settings are summarized in table 5.1.

Table 5.1.: Training settings for the self-supervised pretext task Inpainting-Superresolution

Input size	512x512	Loss Function	MAE + Percept. Loss[59]
Num samples	286477	Optimizer	Adam[105]
Validation	20%	Initialization	Kaiming[40]
Batch size	2	Learning rate	1e-3
Num Epochs	10	Weight Decay	1e-3

For evaluation, the pixel MAE is considered separately from the entire loss function. After 10 epochs, pixel MAE has dropped to 9.1e-3, which is less than 1%, and started stagnating. This indicates that the model is capable of predicting very accurately the RGB values of the original image and thus also filling the black boxes sensibly. However, it cannot be directly concluded if texture also corresponds to the surrounding cloud types. Often, predicting missing parts in images results in blurry areas. This is due to the fact that the average error to target pixels is low for such predictions [57]. Therefore, the perceptual loss [59] was added to increase texture and minimize blurring. To better estimate the ability of also creating sensible texture, inspecting the images and especially the predicted boxes can be instructive.

In figure 5.2, different sample images are depicted showing input, prediction and ground truth. At first sight, there are almost no differences visible between the generated and the target image. Only by taking a closer look, as in figure 5.3, there are artifacts visible in the areas corresponding to the black boxes in the input. Clearly observable is also the increase in resolution compared to the input image. Unlike in the input image, the clouds exhibit sharper contours and more detailed texture.

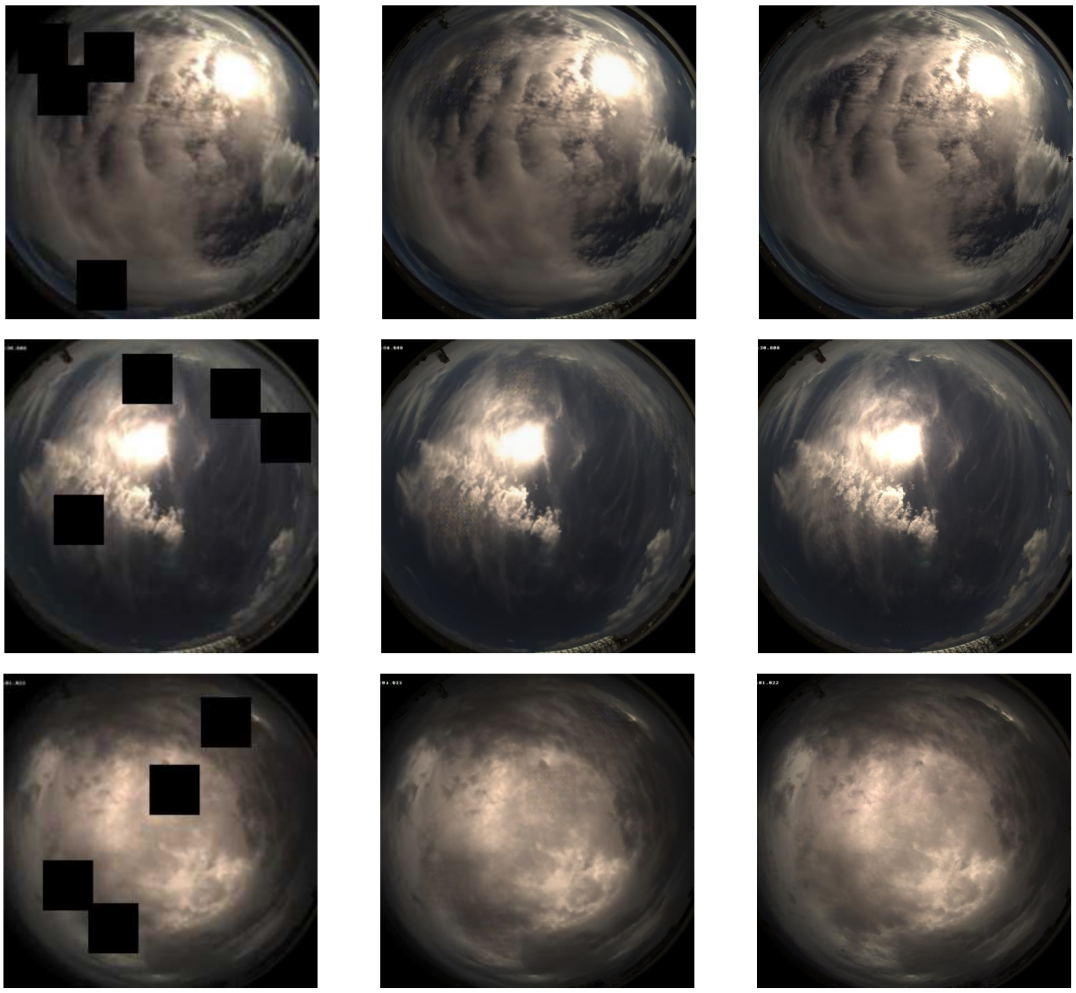


Figure 5.2.: Examples of generated images in Inpainting-Superresolution task. Left: input, middle: prediction, right: ground truth.

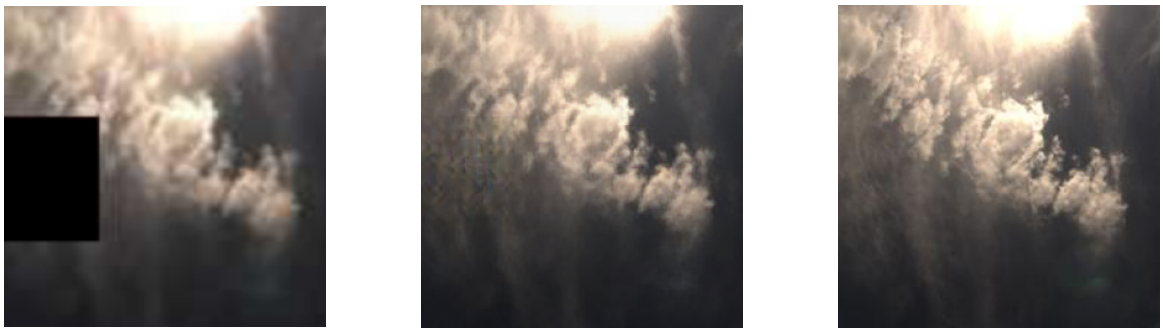


Figure 5.3.: Cutout of 2nd row in figure 5.2. Left-to-right: input, prediction, ground truth.

5.1.2. DeepCluster

For the second pretext task, DeepCluster, a plain ResNet34 architecture was applied to learn representations. Data augmentation and validation was omitted as clustering continuously reassigns the data with other pseudo-labels. Hence, the last layer has to be reinitialized after each reassignment. Consequently, overfitting is already inhibited due to constantly changing pseudo-labels. Furthermore, neither the LR finder nor one-cycle policy were applied, but the hyperparameters were adapted from the original paper [63]. Data normalization corresponds to the Inpainting-Superresolution task, as the dataset is the same (see section 4.4.3).

Here, training was executed for 50 epochs overall. An epoch of clustering and subsequent CNN training is executed in about an hour. The reason why the DC method is much faster in processing an epoch is in particular because of the larger batch size. This comes from the smaller amount of GPU memory that is required. A U-Net architecture, as described in section 5.1.1 has a lot more learnable parameters that need to be loaded into GPU memory. In table 5.2, the training settings for DC are listed.

Table 5.2.: Training settings for the self-supervised pretext task DeepCluster. ImageNet initialization refers to pretrained weights from the ImageNet dataset.

Input size	512x512	Loss Function	Cross Entropy
Num samples	286477	Optimizer	SGDW
Validation	-	Initialization	Kaiming[40]/ImageNet
Batch size	32	Learning rate	5e-2
Num Epochs	50	Weight Decay	1e-5
Clustering	k-means	Num clusters	{30, 100, 1000}

To address the issue of raw color images as input for the DeepCluster CNN, it was also initialized with pretrained ImageNet weights. The advantage is that especially the first layer already contains useful edge and color detectors. As it is going to be discussed in sections 5.2 and 5.3, initializing the DC network with pretrained weights indeed leads to higher accuracies for the classification and segmentation tasks. In total, four training sessions were run, the first with randomly initialized weights and $k = 1000$ clusters, the others with pretrained ImageNet weights and $k = \{30, 100, 1000\}$.

In figure 5.4 exemplary plots of the final cluster assignments after training with pretrained weights and $k = 100$ clusters are shown. Four clusters with four random samples each were chosen randomly. As assumed, clustering incorporates many different features apart from the cloud type. In the top row, rain drops on the camera lens seem to be characteristic for this cluster. For the second row, scattered low-layer clouds covering large areas of the sky are dominant. The next cluster focuses on thin high-layer conditions with rather low sun elevations. In the last row, ASIs with high TL and the sun close to the zenith are depicted. Noteworthy here is the most right image. It appears that turbid atmospheric conditions represents the most relevant feature for this cluster and clouds towards the edge of the image have less influence on cluster assignment.

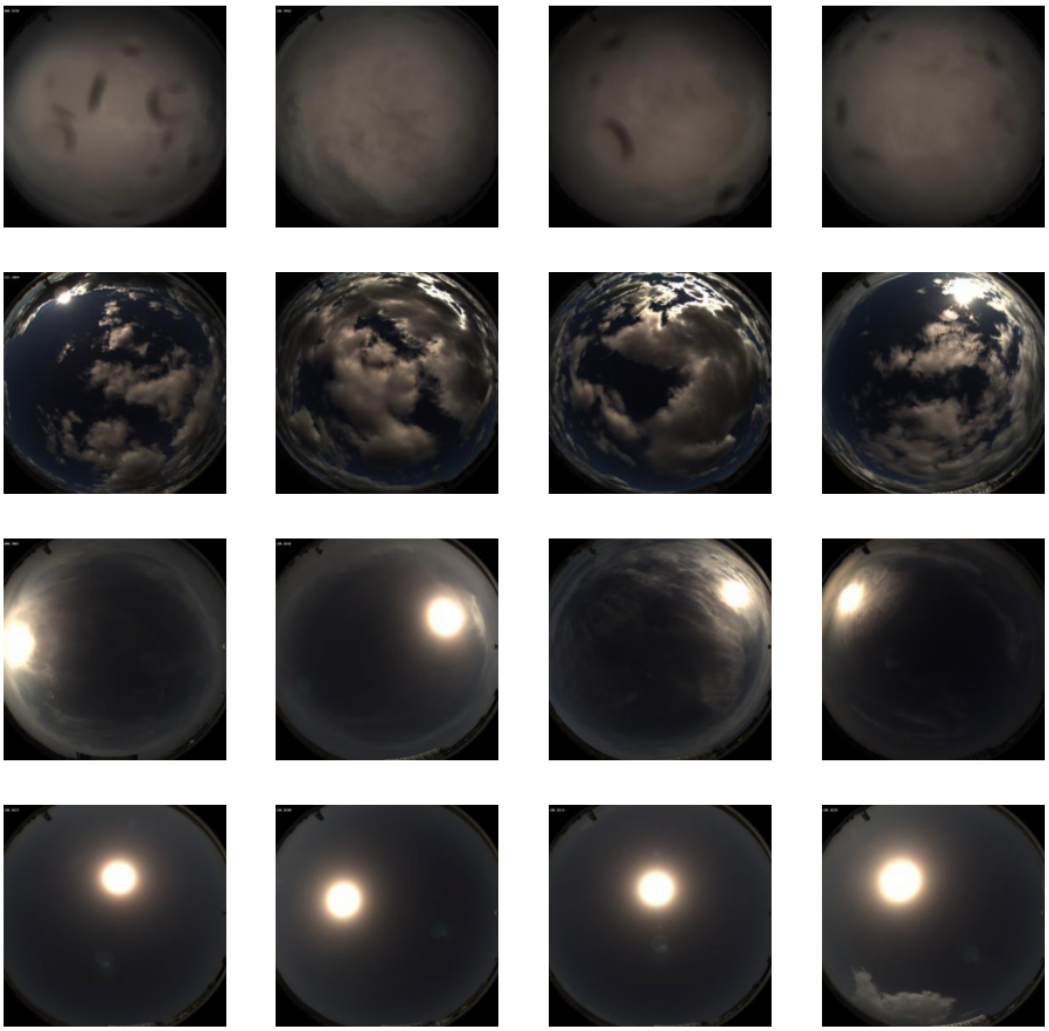


Figure 5.4.: Random selection of cluster assignments for 4 arbitrarily chosen clusters with $k = 100$.

5.2. Evaluation of multi-label classification

Automatically assigning labels to ASI allows to better estimate current weather conditions, and can be beneficial for meterology and climatology. Moreover, accurate statements about the sun are very valuable in the context of solar energy.

This section evaluates the ability to predict labels from ASIs reliably. Of particular interest is the effectiveness of self-supervised pretraining which is compared to random initialization and pretrained ImageNet weights. To further identify challenging conditions, some samples with high loss are inspected.

The models were trained by applying the methods described in section 4.3. A summary of the training settings is presented in table 5.3.

When using pretrained weights, training was started with a frozen network for 20 epochs. Hence, only the FC part was trained and the convolutional part of the network stayed unchanged. Afterwards, it was unfrozen and trained entirely for 10 more epochs. These values for the number of epochs were chosen after preliminary studies with longer training periods. I noticed that the validation loss did not decrease further after 20 epochs with a frozen network. The same applied for fine-tuning the entire model after 10 epochs.

Table 5.3.: Training settings for multi-label classification. When there are two values, the first refers to the frozen state, the latter to the unfrozen one.

Input size	512x512	Loss Function	BCE
Num samples	770	Optimizer	Adam [105]
Validation	20%	Initialization	see 4.3.2
Batch size	32	Learning rate	1e-2, 1e-3
Num Epochs	20, 10	Weight decay	1e-2

As evaluation metrics, accuracy and F1-score were used, defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

Accuracy is the ratio of all true predictions (true positives (TP) and true negatives (TN)) to the sum of all predictions. The F1-score is a common measure for multi-label classification problems, as it includes precision and recall in a single value. While precision takes into account the number of false positives (FP), recall is a measure for the frequency of false negatives (FN). Both are important for classifying ASI, as the model shall not miss a label nor predict it falsely. By applying the default value of 0.5 as threshold, it is determined whether the output score of the logistic function results in a positive prediction of the respective label.

The cloudiness labels *Clear Sky*, *Overcast*, *Cloudy-SV*, *Cloudy-SC* are mutually exclusive by definition. Due to binary cross entropy though, all labels were trained independently such that two cloudiness labels may be predicted simultaneously. To solve that issue, postprocessing is applied on this subset of labels by comparing the output scores and selecting the label with highest value.

For the DeepCluster task, the influence of initialization and k (number of clusters) was evaluated by comparing accuracy and F1-score on the validation set. As depicted in table 5.4, using pretrained weights is beneficial, improving accuracy and F1-score about 0.9% and 1.7% points. Also, smaller k lead to higher accuracy and F1-score. Yet, the differences are less than 1% for both metrics indicating that k plays a minor role for learning representations in ASIs. In the following, only the pretrained DeepCluster model with $k = 30$ (DC-30) is considered.

Table 5.4.: Determining the best DeepCluster model for classification with respect to the number of clusters k used for pretraining.

Metric	Rand Init	ImageNet Init		
	k=1000	k=30	k=100	k=1000
Acc	88.47	**89.77**	89.53	89.37
F1	81.36	**83.64**	83.27	83.01

Inspecting the results from all initialization approaches, none of the models achieve less than 80% in overall accuracy and 70% in F1-score (see table 5.5). Even random weights can be trained to show good performance, however they cannot compete with the pretrained models. Regarding ImageNet and self-supervised initializations, accuracy and F1-score are around 89% and 83% respectively with the best ones obtained by the DeepCluster method (89.77% and 83.64%). Remarkable was that unfreezing the network did not improve the performance significantly. Already after training for 20 epochs with a frozen network, the accuracies were around 88% for all but the random initialized model which was never frozen.

Table 5.5.: Results of multi-label classification with different initializations on validation set (154 images). Metrics are accuracy and F1-score. IP-SR: Inpainting-Superresolution. DC: DeepCluster

Label	Random		ImageNet		IP-SR		DC	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Total	81.74	70.36	89.45	83.33	89.61	83.46	**89.77**	**83.64**
Clear Sky	93.51	72.22	98.05	91.43	98.05	90.91	**99.35**	**97.14**
Overcast	91.56	71.11	94.16	79.07	**96.10**	**86.96**	94.16	80.00
Cloudy-SV	83.77	80.62	94.16	93.53	91.56	90.65	**94.81**	**94.20**
Cloudy-SC	84.42	75.51	**91.56**	**85.71**	90.91	84.44	90.91	84.44
Low-Layer	70.78	74.29	81.17	83.62	**82.47**	**84.21**	81.82	82.72
Mid-Layer	67.53	48.98	77.92	67.31	**81.82**	**73.08**	79.22	70.91
High-Layer	77.92	58.54	83.77	74.23	83.12	71.74	**86.36**	**77.42**
Sun-Free	84.42	75.00	**94.81**	**91.49**	92.86	88.89	91.56	86.60

Taking a look at some of the positive predictions reveals that also multi-layer conditions and clouds with lower PZA can be detected correctly. In figure 5.5a, opaque low-layer clouds cover the sun. Although they are distorted due to the proximity to the horizon, the model correctly classifies them as low-layer. An advantage in this particular ASI is the low turbidity making the visible clouds very salient. The examples 5.5b, 5.5c show that the model is capable of correctly recognizing cloud types also under more turbid conditions. For the latter, the model also noticed the presence of low-layer clouds overlapping with the high-layer sheet. In the right bottom image 5.5d the model even correctly identified all layers.

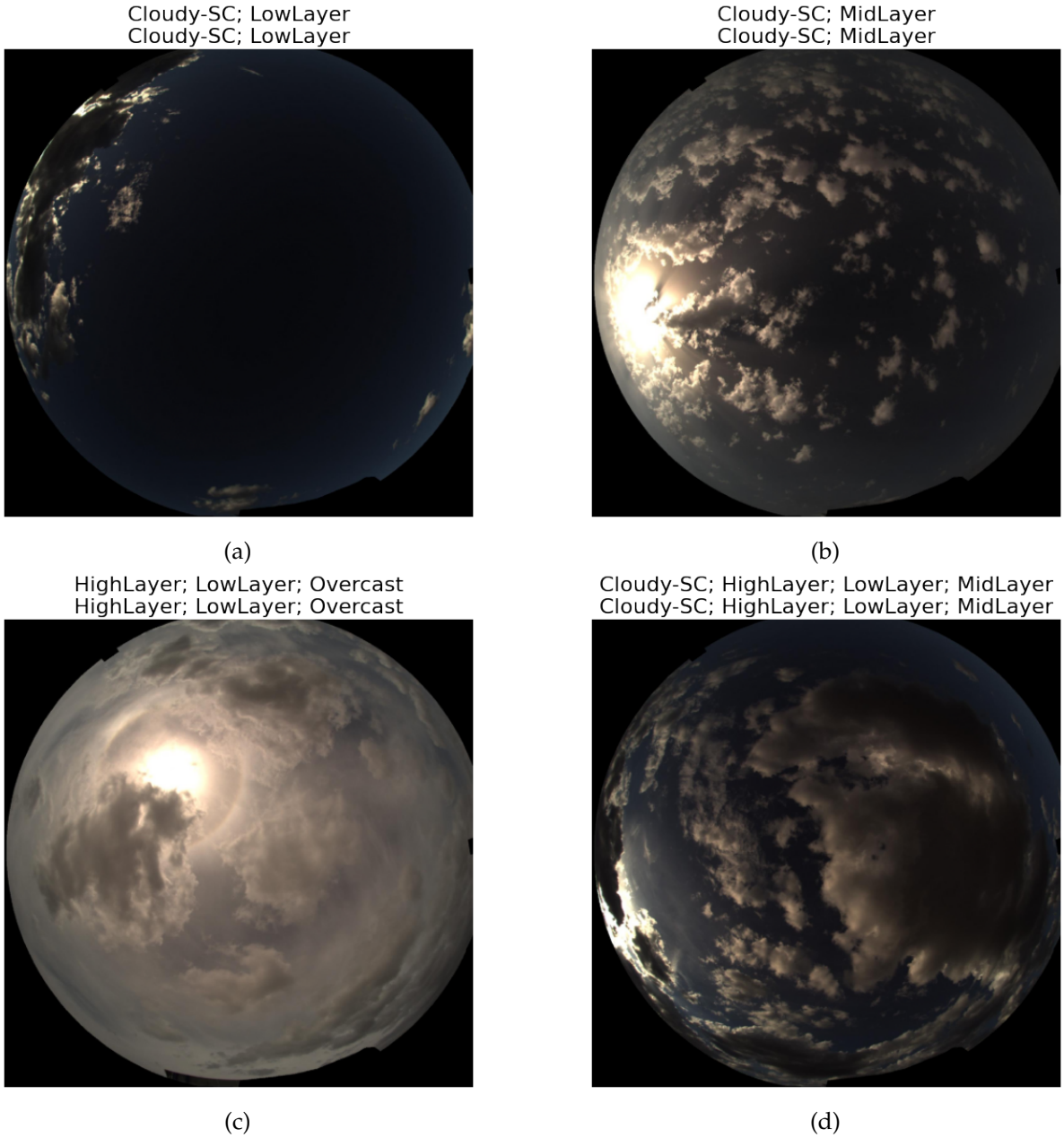


Figure 5.5.: Four examples of correct predictions. First row are single layer conditions, second are multi-layer conditions. Upper heading is ground truth, below prediction.

Confusion matrices are prevalent tools to visually examine the strengths of a model. In classification, the number of predicted classes is counted with respect to the actual class and depicted row- or column-wise. As a result, true predictions are shown in the diagonal. Since multi-label classification is applied, the labels need to be split into mutually exclusive groups and considered separately. Therefore, three categories are defined: *cloudiness*, *cloud-layers*, and *sun free*. For the cloud-layer group, the combination of labels in each ASI is considered as a separate label. In this context, a prediction is only true if all cloud-layer labels are detected correctly. This leads to a seemingly worse performance than indicated by the accuracy/F1-score.

The quality of the confusion matrices for all initializations is similar which is why only the best model (DC-30) is examined in the following. For completeness, the confusion matrices of all other approaches can be found in appendix A.1.

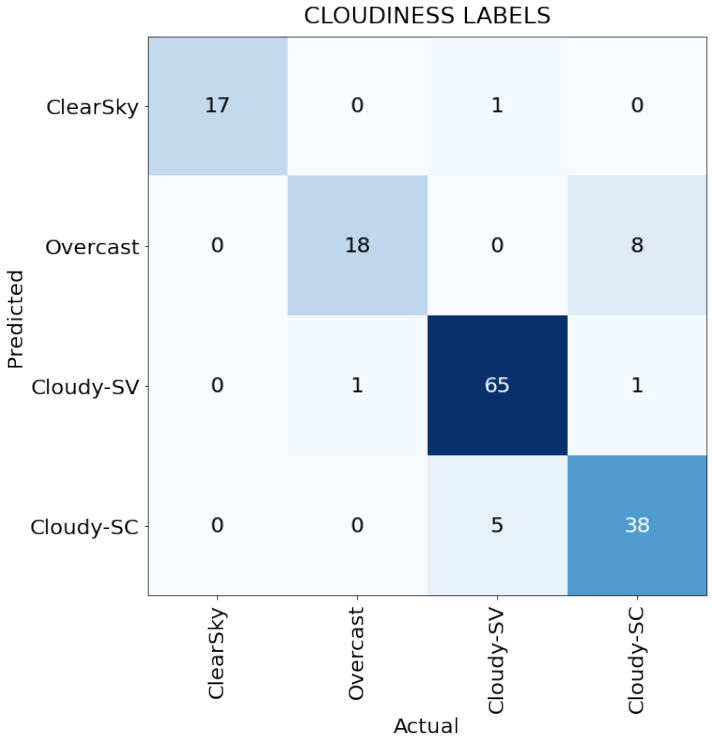
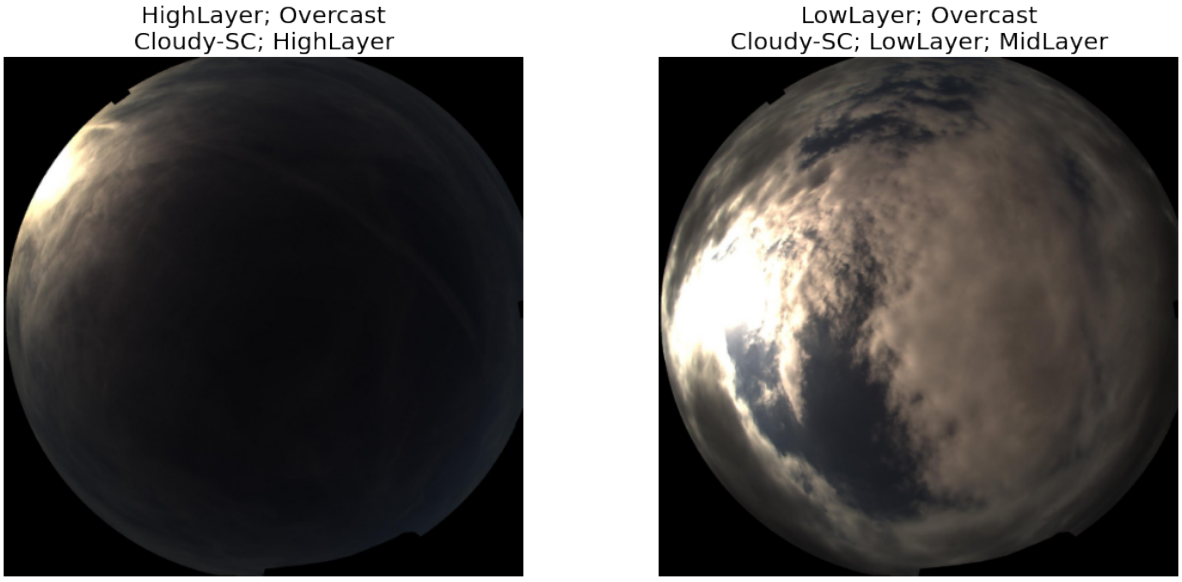


Figure 5.6.: Confusion matrix for label groups cloudiness.

Regarding the confusion matrix for cloudiness in figure 5.6, a clear diagonal can be seen. Significant discrepancy is only present for the predicted-actual pairs *Overcast - Cloudy-SC* and *Cloudy-SC - Cloudy-SV*. These confusions result from smooth transitions between cloudiness labels. For example, overcast is not strictly defined as 100% cloud coverage. In figure 5.7a, a misclassification may come from smaller cloudless parts. Another difficulty is due to thin high-layer conditions as in figure 5.7b, especially for low sun elevations. Similarly, there is no clear threshold whether the sun disk is visible or not. Hence, it is important to distinguish between misclassifications that are due to ambiguous conditions and clearly incorrect ones.



(a) (b)

Figure 5.7.: Examples of false cloudiness predictions. Smooth transition between labels can lead to misclassification. Upper heading is ground truth, below prediction.

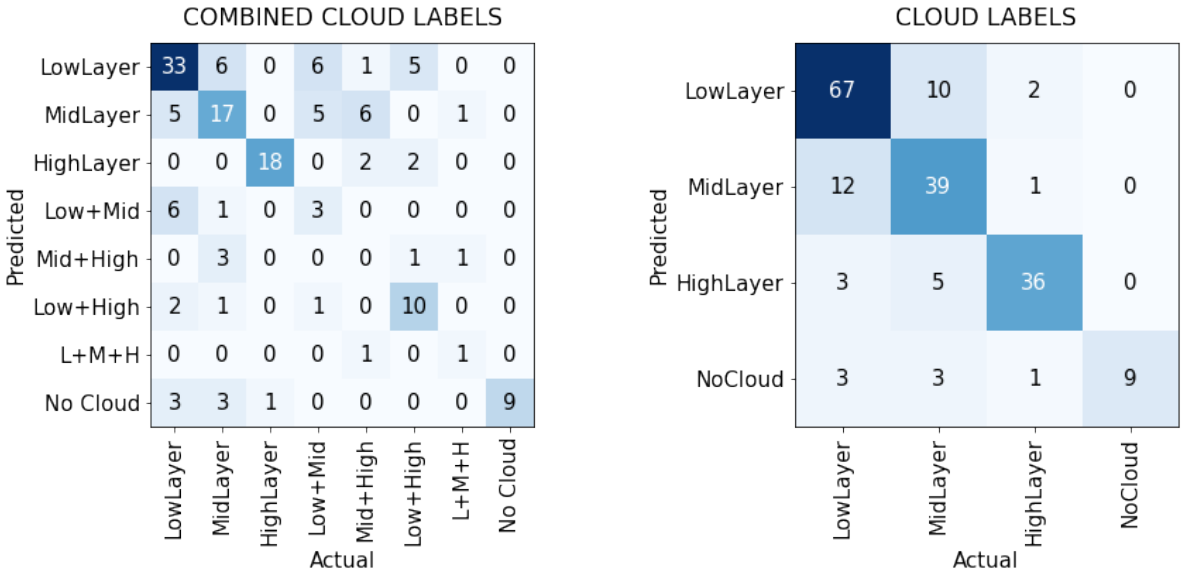
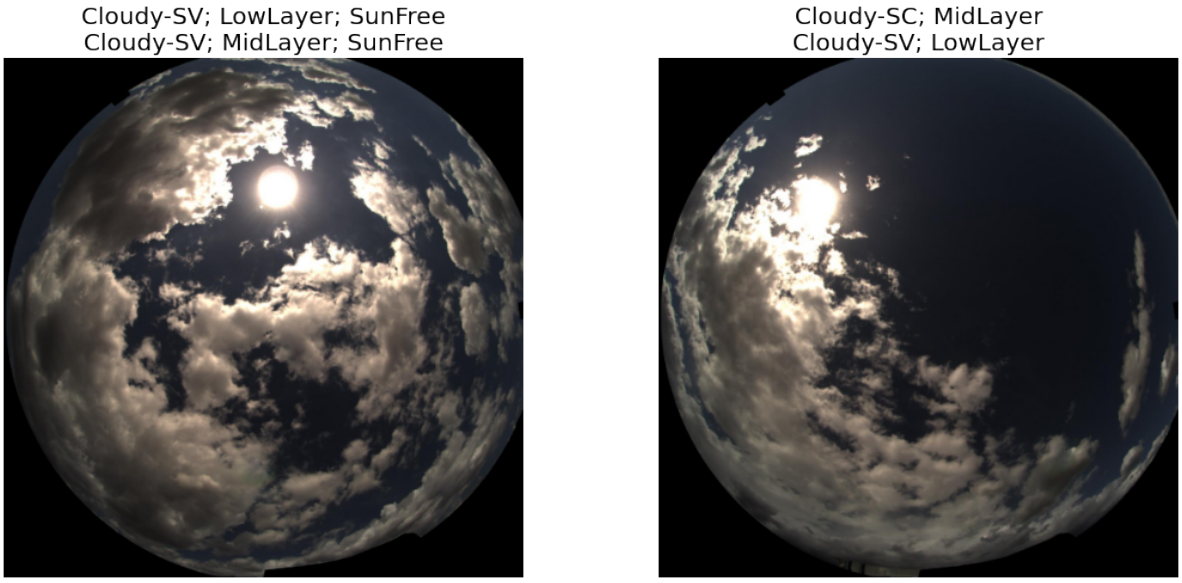


Figure 5.8.: Confusion matrices for label group cloud-layers. Left: Combinations of labels are regarded separately. Right: Each label is considered independently.

In terms of cloud-layer labels, there is more deviation from the diagonal in the confusion matrix, particularly if combinations are considered separately (see figure 5.8). One explanation is the large variation in appearances for each cloud type. In the training set, only a small fraction of possibilities is represented. On closer inspection however, many predictions are partially correct. For multi-layer conditions, one label is always predicted correctly. In single-layer situations, sometimes a second layer is falsely detected, but the true one is identified most of the times. On the right side of figure 5.8, the labels are considered individually. Here, the diagonal is clearly recognizable. It represents the TPs of each label whereas each misclassified or non-detected label counts to the respective FPs/FNs. These results are consistent with the accuracies and F1-scores for cloud-layer labels in table 5.5 which are between 79-86% and 70-82% respectively for the DC-30 model. A significant confusion is in particular between low- and mid-layer clouds. When inspecting some false predictions (figure 5.9), it becomes evident that there are cases that are also not easy to classify for humans. Overlapping cloud layers, broad twilight zones, low sun elevations and camera distortion are all influencing factors that make classification so challenging.



(a) (b)

Figure 5.9.: Examples of false cloud-layer predictions. Low- and mid-layer clouds can have similar appearances. Upper heading is ground truth, below prediction.

Finally, the capability of predicting the label *sun free* is analyzed. Achieving an accuracy of 91.56%, there are still 10 FP and 3 FN samples out of 109 sun free situations (see figure 5.10). There are two main reasons for misclassifications. Either thin high-layer clouds cover the sun or parts of the twilight zone spread out to the sun disk. In both cases, these factors are not recognized as decisive features for the *sun free* label. A simple way to increase precision is by raising the threshold for the *sun free* label such that higher scores are required to indicate a free sun disk. When set to 0.8, the number of FP is reduced to 3 while having 7 FN.

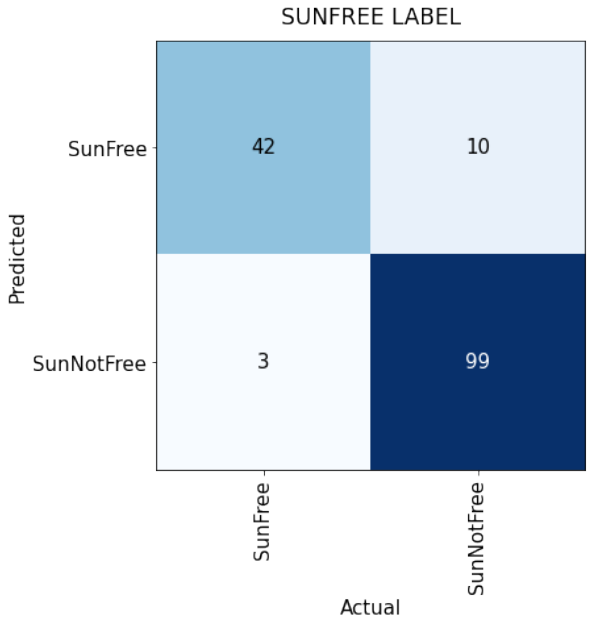
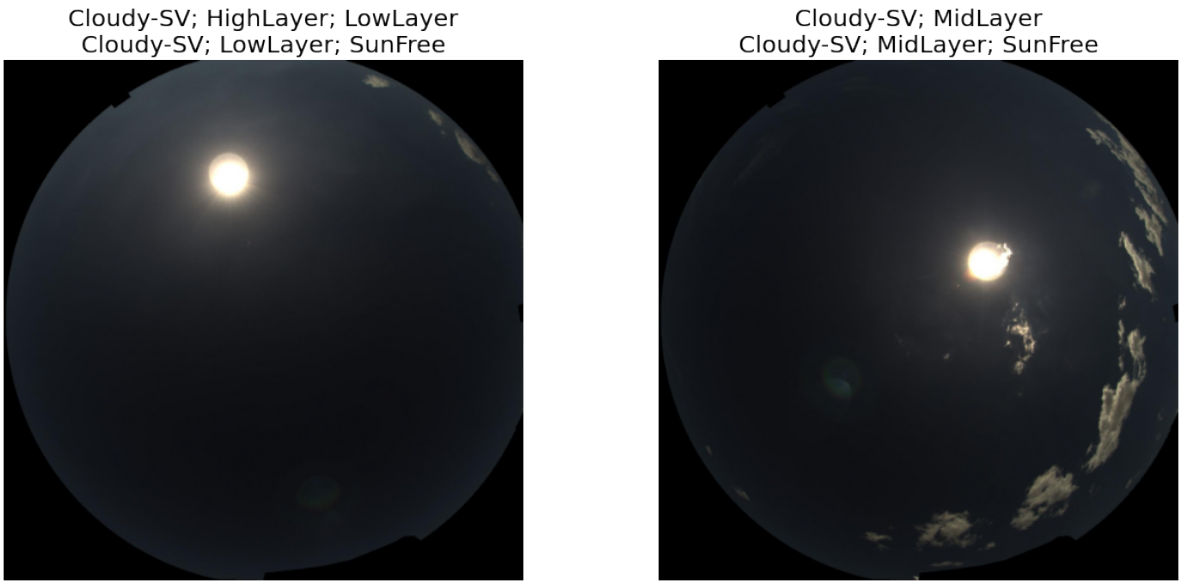


Figure 5.10.: Confusion matrix for sun free label.



(a) (b)

Figure 5.11.: Examples of false sun free predictions. Upper heading is ground truth, below prediction.

Overall, the results of multi-label classification are very promising, especially in consideration of the limited supervised dataset. However, there is also room for improvement, in particular for detecting cloud-layers. To achieve this, a larger annotated dataset would be useful. Particularly important for smaller datasets is that the amount of incorrectly labeled images is reduced to a minimum, as few images can already have a negative effect on learning. Moreover, the potential of self-supervision has not been exploited entirely and further studies need to be made, evaluating other pretext tasks and larger datasets.

5.3. Evaluation of semantic segmentation

Segmentation of ASIs is a research field which has been gaining a lot of interest lately. Because of the complexity of cloud formations in the sky, even humans require a lot of expertise to classify diverse situations. As a result, most state-of-the-art methods only perform binary segmentation. Nonetheless, there is a substantial need for more fine-grained cloud classification in ASIs on pixel-level to account for independent cloud layers with distinct motion vectors and optical properties.

In the following section, the four-class segmentation approach using deep learning techniques is evaluated. As for multi-label classification, the performance with different initializations is compared.

5.3.1. Benchmark of different initializations

The training setup is similar to the multi-label classification. A detailed description of the applied methods can be found again in section 4.3. A summary of these settings is presented in table 5.6. For initializations with pretrained weights, training started with a frozen network for 20 epochs. Fine-tuning of the entire network was then applied for another 20 epochs. It was trained for more epochs than in classification because it took longer for the validation loss to reach a minimum value.

Table 5.6.: Training settings for semantic segmentation. When there are two values, the first refers to the frozen state, the latter to the unfrozen one.

Input size	512x512	Loss Function	Cross Entropy
Num samples	770	Optimizer	Adam [105]
Validation	20%	Initialization	see 4.3.2
Batch size	4	Learning rate	1e-3, 1e-4
Num Epochs	20, 20	Weight Decay	1e-2

Pixel-accuracy is used to estimate overall performance on the entire ASI. Furthermore, mean IoU is determined which is an average of the class-specific IoUs and a common measure in segmentation tasks. Apart from IoU, precision and recall are also evaluated to examine class-wise performance. The definitions of precision, recall and accuracy are the same as in

section 5.2. IoU is the ratio of the intersection of prediction and target to the union of both. Mathematically, it is defined as follows:

$$IoU = \frac{TP}{TP + FP + FN}$$

Here, TP indicates the sum of pixels for which a specific class was predicted correctly. Analogously, this holds for TN, FN and FP. The border part of an ASI is not of interest, so it is not included for computing the metrics.

Again, the best model from the DeepCluster pretraining is determined first. Like for multi-label classification, the model trained with $k = 30$ clusters (DC-30) performs best (see table 5.7).

Table 5.7.: Determining the best DeepCluster model for segmentation with respect to the number of clusters k used for pretraining.

Metric	Rand Init	ImageNet Init		
	k=1000	k=30	k=100	k=1000
Pixel-Acc	84.60	**85.22**	84.86	84.88
Mean IoU	80.09	**80.48**	80.41	80.22

In comparison with the results from the other initializations, DC-30 also achieves the best overall pixel-accuracy (table 5.8). Regarding mean IoU, this time the IP-SR model gets slightly better results. Noteworthy is a significant improvement compared to ImageNet initialization for both self-supervision methods, gaining 3% points for accuracy and mean IoU and thus indicating the benefits of this technique. Random initialization is even outperformed by roughly 7% points.

Table 5.8.: Results of semantic segmentation with different initializations on validation set (154 images). Metrics are pixel-accuracy and mean IoU. IP-SR: Inpainting-Superresolution. DC: DeepCluster

Metric	Random	ImageNet	IP-SR	DC
Pixel-Acc	78.34	82.05	85.09	**85.22**
Mean IoU	72.11	77.10	**80.58**	80.48

Overall, the self-supervised models also achieve similar improvements with respect to each class as depicted in table 5.9. To compute the average values for a class over all images, each sample is weighted. As clouds can cover just few pixels or the entire ASI, considering spatial extents of clouds results in more meaningful average values. Hence for each sample, precision, recall or union is scaled by prediction, target or union size respectively. For example, the resulting equation for the average IoU for class c and N sample images is:

$$IoU_c = \sum_i^N a_{c,i} \cdot \frac{TP_{c,i}}{TP_{c,i} + FP_{c,i} + FN_{c,i}} = \frac{\sum_i^N TP_{c,i}}{\sum_i^N TP_{c,i} + FP_{c,i} + FN_{c,i}}$$

$$a_{c,i} = \frac{TP_{c,i} + FP_{c,i} + FN_{c,i}}{\sum_i^N TP_{c,i} + FP_{c,i} + FN_{c,i}}$$

Table 5.9.: Class-wise results of semantic segmentation with different initializations on validation set (154 images). Metrics are precision, recall and IoU in %. IP-SR: Inpainting-Superresolution. DC: DeepCluster

	Class	Random	ImageNet	IP-SR	DC
Precision	Sky	90.92	93.67	**94.35**	94.25
	Low-Layer	63.61	69.70	**75.58**	73.70
	Mid-Layer	49.14	56.34	67.23	**75.09**
	High-Layer	48.72	58.67	**65.85**	64.73
Recall	Sky	**97.95**	97.53	97.09	97.13
	Low-Layer	71.43	78.74	84.91	**89.13**
	Mid-Layer	26.21	32.38	**46.90**	40.15
	High-Layer	47.32	65.06	67.71	**70.65**
IoU	Sky	89.23	91.50	**91.75**	91.69
	Low-Layer	50.71	58.66	66.63	**67.63**
	Mid-Layer	20.62	25.88	**38.17**	35.43
	High-Layer	31.59	44.61	50.12	**51.02**

The advantage of self-supervision over random and ImageNet initialization is clearly evident here. For all classes, precision, recall and IoU are highest for the self-supervised models, except for recall of *sky*. In all other cases, IP-SR and DC initialization achieve much better results. Unlike for classification, the differences to pretraining on ImageNet are often very significant, especially with respect to cloud types. For the mid-layer class, the DC method reaches about 8% points more for recall, 10% for IoU and almost 20% for precision. Similar improvement due to self-supervision is obtained for the other two cloud types. Surprisingly, the randomly initialized model reaches the highest score for recall of *sky*, but for all models it is higher than 97%. When looking at precision and IoU for *sky*, random initialization leads to much lower scores, indicating an overestimation of that class. Thus, the model detects most of the actual *sky* pixels but also predicts many cloud pixels incorrectly as *sky* which is undesirable for nowcasting systems. Taking this into account, the self-supervised models perform best for all classes.

Looking at some predictions in figure 5.12, it can be seen that the model is capable of producing sensible segmentation masks even for difficult multi-layer conditions. However, there are also cases where a proper distinction between cloud types does not work (figure 5.13). It seems that especially stratus-like clouds covering large fractions of the hemisphere are often not easy to identify. Missing texture and structure of those clouds makes it particularly hard to infer the correct class. As they can occur in every layer such conditions are one of the most difficult ones.

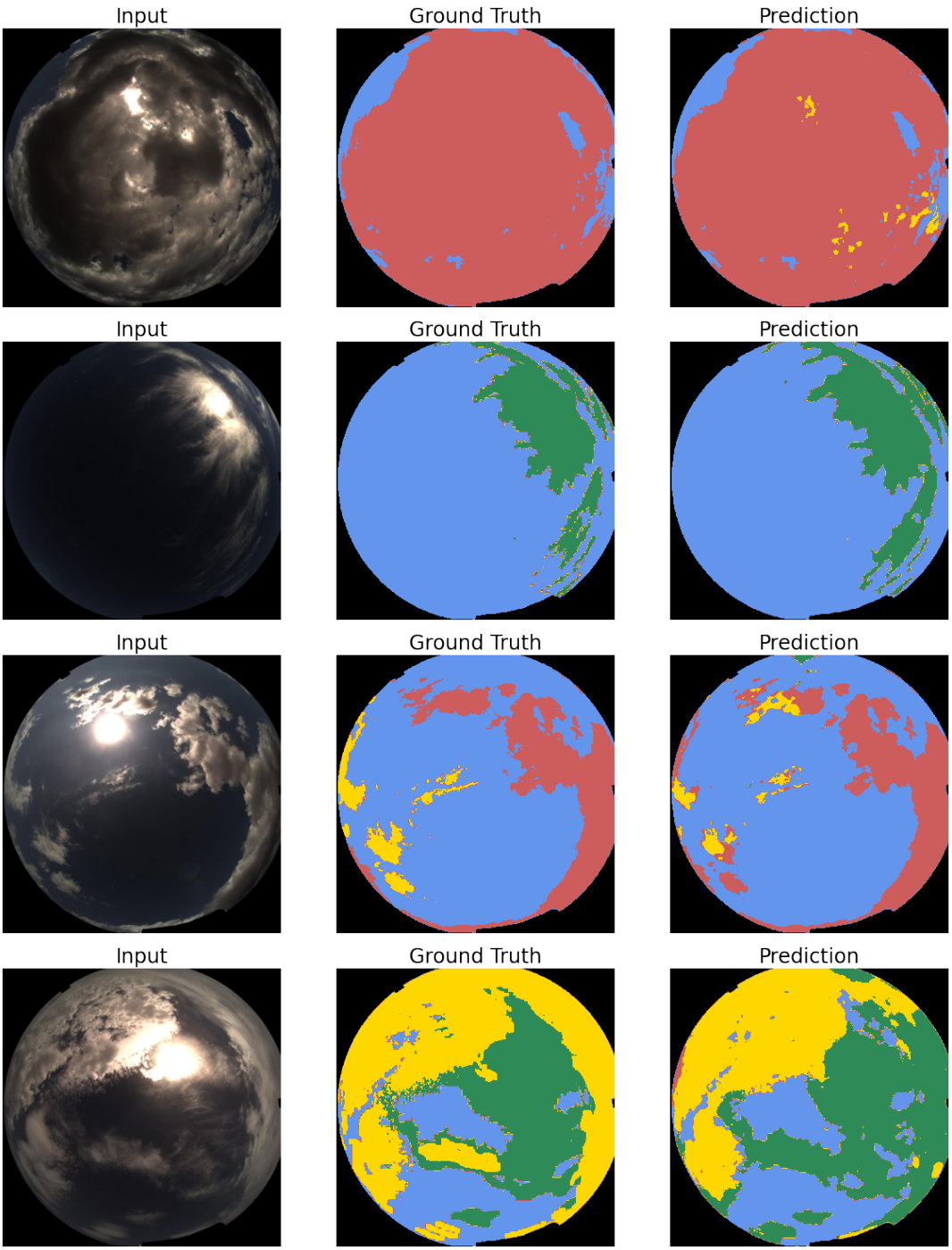


Figure 5.12.: Examples of positive segmentation results for single- and multi-layer conditions (blue: sky, red: low-layer, yellow: mid-layer, green: high-layer).

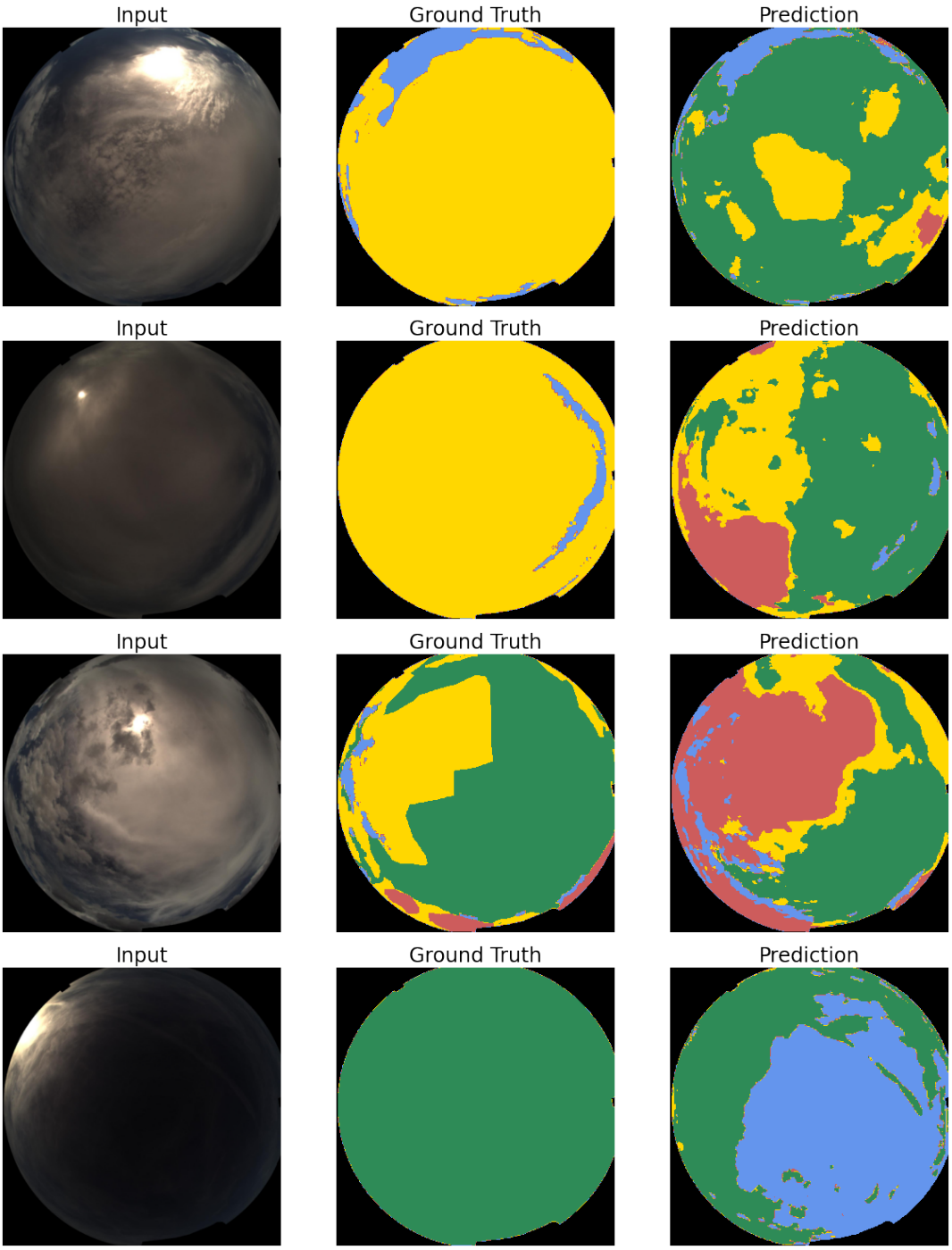


Figure 5.13.: Examples of negative segmentation results. Especially mid-layer and overlapping multi-layer clouds are segmented poorly. (blue: sky, red: low-layer, yellow: mid-layer, green: high-layer).

Considering the scores for the mid-layer class, low recall and IoU values compared to rather high precision are noticeable. This indicates that mid-layer clouds are often not detected as such. But when they are predicted, it is likely that the prediction is correct. In general, the distinction of low- and mid-layer clouds is challenging. Due to varying vertical thickness they can have similar appearances. Additionally, twilight zones of low-layer clouds often visually resemble mid-layer clouds. For example, in the first and third row of figure 5.12, brighter parts of the low-layer clouds are assumed to be mid-layer ones.

Distortion effects from the fish-eye lens and low resolutions of distant clouds further impede recognizing important features. Thus, areas closer to the horizon are also misclassified more frequently.

To quantitatively analyze misclassifications, confusion matrices are again a useful tool. In the following, only the confusion matrix of the DC-30 model is considered. A full comparison of all initializations is presented in the appendix A.2.

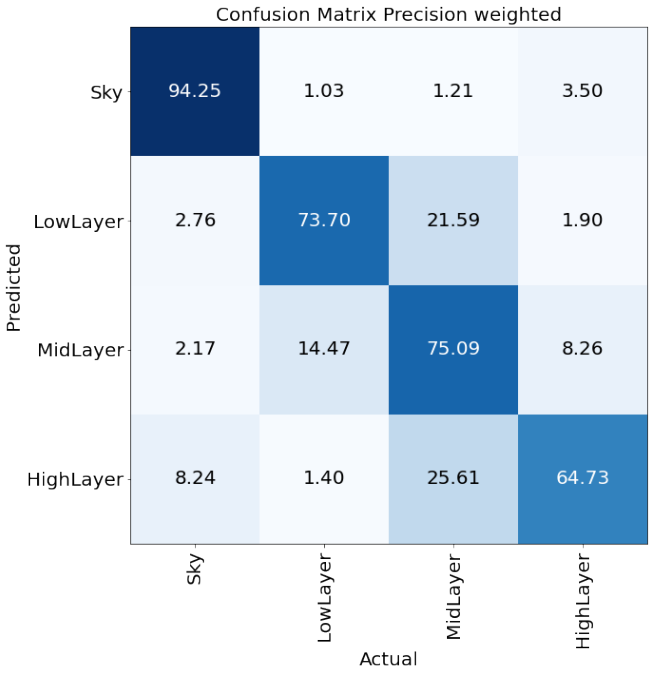


Figure 5.14.: Confusion matrix for average precision of the DC-30 model.

For precision, a clear diagonal is visible (figure 5.14). Each predicted class is mixed up mainly with one other class. A bit more than one fifth of the predicted low-layer pixels were actually labeled as mid-layer. For predicted high-layer pixels, approximately one quarter is mid-layer. In terms of predicted mid-layer, the majority of 75% is correct and most frequent confusion is with low-layer clouds (̃15%). Hence, misclassifications are mainly within adjacent cloud layers.

Regarding Recall (figure 5.15), major deviations from the diagonal are primarily for the mid-layer class. As previously stated, all models have difficulties to detect mid-layer clouds reliably, and often classify them as low- or high-layer clouds. The only other high value off the

diagonal is predicted sky that is actually high-layer. Almost 20% of the actual high-layer pixels are falsely predicted as sky. One reason for this substantial proportion are thin high-layer clouds covering large parts of the hemisphere. As can be seen in the last example of figure 5.13, the overcast caused by high-layer clouds is not recognized leading to a large amount of misclassified pixels. As the overall percentage of high-layer pixels is only about 10%, such images have significant influence on the recall.

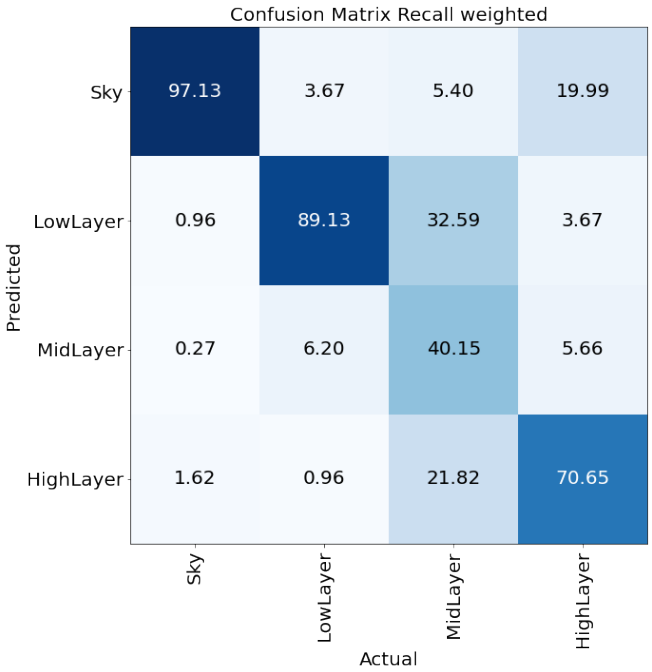


Figure 5.15.: Confusion matrix for average recall of the DC-30 model.

Finally, the IoU quantifies the overlap of the prediction and ground truth mask of a certain class. Although the absolute values of the diagonal are lower, the qualitative results are good. There are no frequent and spatially extensive overlaps of falsely predicted classes and actual ground truth. Only for the mid-layer class the IoU with respect to predicted low-layer and high-layer is above 10%. To address this issue, further research is required to train models that learn to better distinguish between these cloud types. Still, the results are very valuable as the distinction of visually opaque low-layer and more transparent high-layer clouds works well. Moreover, the mere differentiation between clouds and clear sky is one of the most important aspect that has not been analyzed for the models so far.

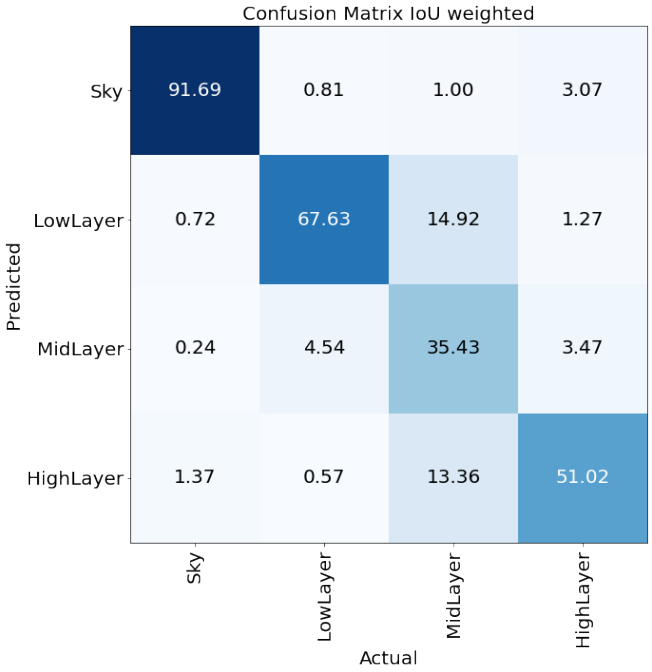


Figure 5.16.: Confusion matrix for average IoU of the DC-30 model.

5.3.2. Training a binary Model vs. postprocessing the 4-Class model

To evaluate binary segmentation, it is first examined if retraining with a binary ground truth leads to better performance. For retraining a binary model, architecture and training settings stay the same. The only difference is that the ground truth used for training consists of a single cloud class. The binary mask for this class is created by taking the union of the low-, mid and high-layer pixels.

The alternative approach to obtain binary segmentation masks is to use the 4-class model and take the union of the predicted masks. The values for precision, recall and IoU are briefly compared to assess the benefit of training a binary model. For both methods the self-supervised DC-30 model is used.

As shown in table 5.10, the differences for all metrics are small. But the retrained binary model achieves better results in general. The most significant improvement is in IoU for the cloud class which is almost 1% higher. Also, the overall pixel-accuracy is slightly higher with 95.15% for retraining and 94.82% for postprocessing.

Nonetheless, both approaches achieve excellent results in binary segmentation that are competitive to state-of-the-art methods. In the recent benchmark [91], the best method evaluated on the entire ASI achieved 92%. Regarding the paper [94] which presents the first pixel-level cloud classification, the average accuracy on their dataset for binary segmentation is specified as 93.71%. However, such comparisons can only be interpreted as rough assessments as performance strongly depends on utilized data and corresponding conditions. For validation I thus compare the deep learning approach with a CSL [81] on the dataset presented here.

Table 5.10.: Evaluation of training a model with binary ground truth by comparing accuracy, precision, recall and IoU to postprocessed results of the 4-class model.

Metric	Class	Postproc.	Retrained Model
Accuracy	-	94.82	**95.15**
Precision	Sky	94.25	**94.76**
	Cloud	95.70	**95.75**
Recall	Sky	97.13	**97.14**
	Cloud	91.52	**92.31**
IoU	Sky	91.69	**92.18**
	Cloud	87.90	**88.86**

5.3.3. Validating the deep learning model against a CSL

Determining the performance of a state-of-the-art method on the fixed validation set of the annotated data allows conclusive validation of my results. Clear Sky Librarys are regareded as one of the best methods for binary segmentation in ASIs which are also applied in current nowcasting systems, like the one being developed at DLR [81]. Details of the utilized CSL can be found in section 2.2.2.

First, overall accuracy is determined, as well as precision, recall and IoU for the classes *cloud* and *sky* (see table 5.11).

Table 5.11.: Evaluation of CNN (DC-30) and CSL performance by comparing weighted average of precision, recall and IoU.

Metric	Class	CNN	CSL
Accuracy	-	**95.15**	87.88
Precision	Sky	**94.76**	86.82
	Cloud	**95.75**	85.50
Recall	Sky	**97.14**	93.84
	Cloud	**92.31**	79.67
IoU	Sky	**92.18**	82.15
	Cloud	**88.86**	73.24

For all metrics, the DC-30 CNN achieves much better results. Regarding overall accuracy, it outperforms the CSL by more than 7% points which represents a significant amount. Also for all other metrics the improvement is high. In particular for the *cloud* class, the CNN reaches always at least 10% points more than the CSL.

To further assess strengths and weaknesses of the CSL compared to the CNN, segmentation results are evaluated under various conditions. These conditions are defined by the ground truth labels of the multi-label classification. In figure 5.17, a qualitative comparison of both methods is shown.

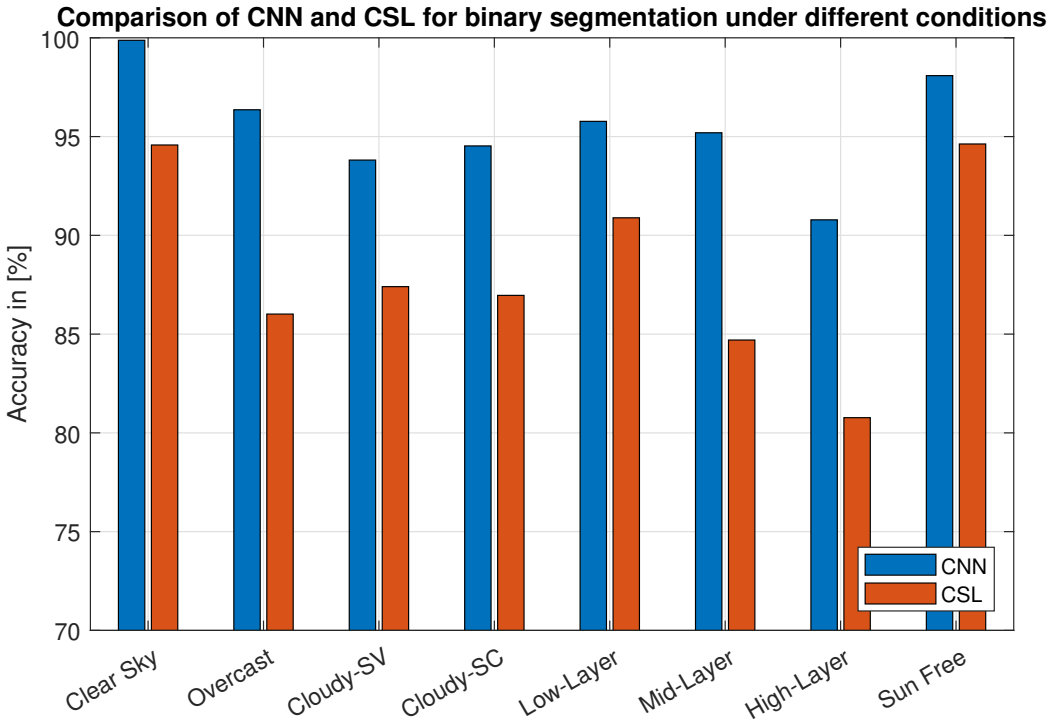


Figure 5.17.: Comparing average accuracy of CNN and CSL for different conditions that were determined by the labels from classification.

Each bar indicates the average accuracy of the CNN and the CSL for images that were assigned with the respective label. As already indicated by clearly lower values for the *cloud* class in table 5.11, the differences are particularly high for conditions with large cloud coverage. Furthermore, especially high- and mid-layer clouds are much more difficult to detect for the CSL which may be caused by similarities of these clouds with other aerosols.

To get a better understanding why the CNN achieves better scores, taking a look at some examples can be helpful. Figure 5.18 shows situations where the prediction of the CNN and the CSL diverge widely. In particular, in the second row the result from the CSL is almost the inversion of the actual mask which may result from inaccuracies in the library itself. However, there are still cases where both models fail to detect thin high-layer clouds, as depicted in the third row. For few images, the CSL beats the CNN. For example in the last row, the sky is covered by a fine layer of Cirrus clouds. Due to low sun elevation, the image appears very dark and the clouds are mainly recognizable close to the sun. Only by zooming into the image, the overcast becomes clear.

Overall, the deep learning approach outperforms the CSL by a large margin. Especially, circumsolar regions that are error-prone to threshold-based methods can be detected more accurately with the CNN. This is probably the most significant advantage in terms of energy-related applications. ASI segmentation usually serves as a basis for nowcasting systems, thus it is crucial that the area around the sun is detected accurately to include their influence on

irradiance. Furthermore, the deep learning model does not require intensive long-term data acquisition to create a camera dependent library. However, the applicability of these CNNs on different cameras and observation sites as well as the need for fine-tuning remains an open question that needs to be addressed in future work.

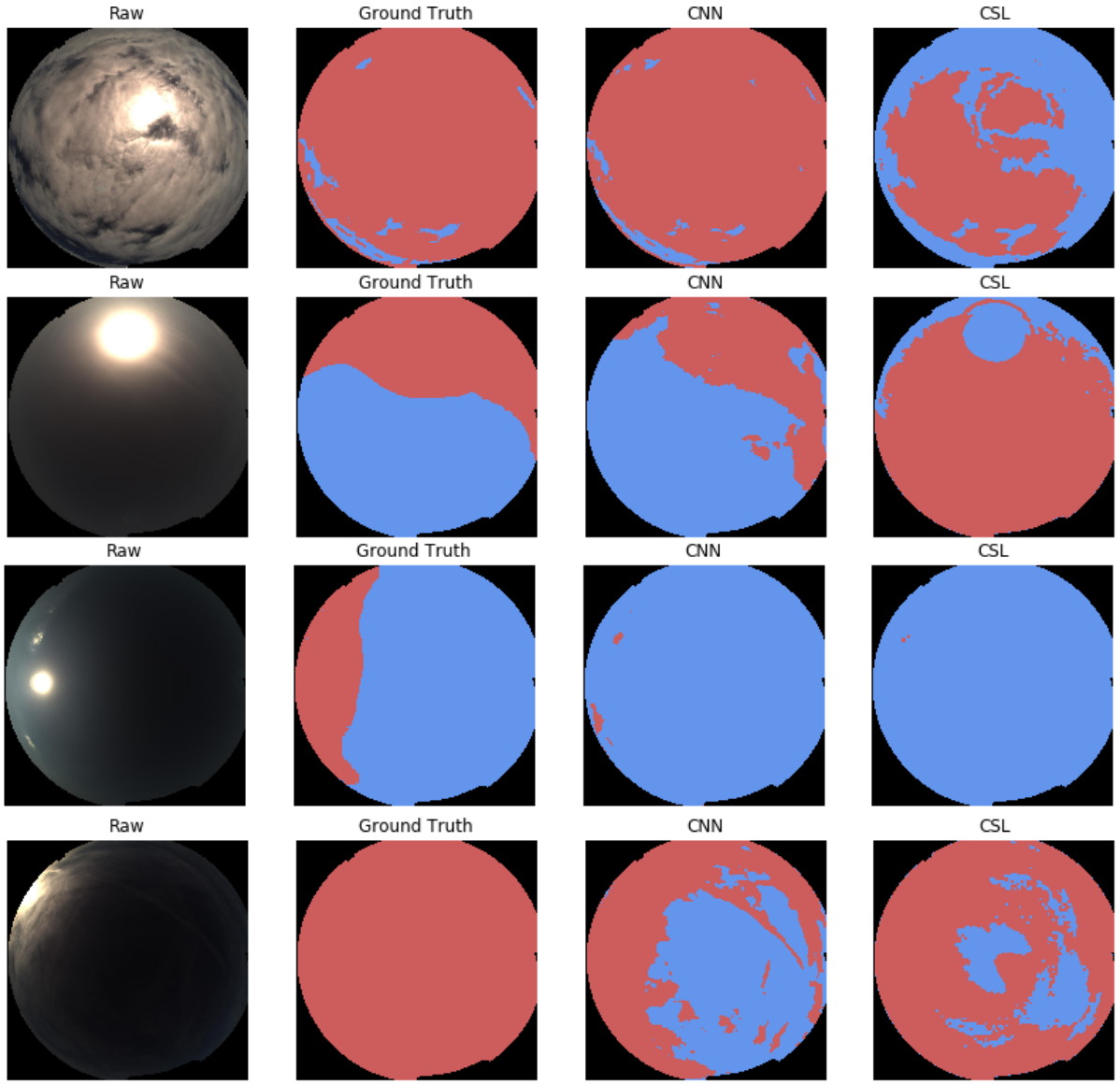


Figure 5.18.: Exemplary results for binary segmentation of CSL and CNN.

6. Conclusion and outlook

In this thesis deep learning techniques for detecting clouds in ASIs were examined. The main contributions are:

- A new dataset containing labels and pixel-wise annotations for 770 images.
- A multi-label classification model based on a ResNet34 architecture to recognize global cloudiness characteristics.
- A semantic segmentation model based on the U-Net architecture that classifies ASIs pixel-wise into four classes.
- Studies on the effectiveness of self-supervised pretraining for ASIs by implementing two pretext tasks (IP-SR, DC) and evaluating the performance on classification and segmentation. A comparison to pretrained ImageNet weights and random initialization was conducted for validation.

For multi-label classification, eight labels were defined: *clear sky, overcast, cloudy-sv, cloudy-sc, sun free, low-layer, mid-layer, high-layer* where multiples can be valid for each image. The best model was pretrained with the DC method and achieved an accuracy of 89.77% and an F1-score of 83.64%. Similar results with an accuracy above 89% and an F1-score above 83% were attained when utilizing pretrained weights from IP-SR or ImageNet. Only random initialization lead to substantially inferior results. For this task, no significant improvements could be noted due to self-supervision compared to ImageNet pretraining. It seems that early layers of a pretrained ImageNet model already yield useful representations that can compete with the self-supervised methods. Still, the results show that a deep learning approach works well for classification even if the dataset is limited. In particular the labels describing overall cloudiness (*clear sky, overcast, cloudy-sv, cloudy-sc*), and *sun free* are predicted very accurately. For each of them, accuracies above 90% were achieved, the best reaching 99.35% (*clear sky*). Misclassifications are primarily due to smooth transitions between these labels. More challenging was the recognition of all visible cloud layers. Although one layer is detected correctly for most predictions, often another was missing or falsely detected. This reflects the difficulty of this task which lies in the high variability and similarity of clouds. Moreover, lower spatial resolution, more profound image distortion and vignetting near the horizon contribute to higher error rates in these image regions.

In semantic segmentation, the four classes correspond to the labels *low-*, *mid-*, *high-layer* and (cloudless) *sky*. This time, a clear advantage of the self-supervised pretraining is recognizable. Achieving pixel-accuracies over 85% and mean IoU over 80%, the DC and IP-SR models outperform the pretrained ImageNet initialization by 3% points in both metrics. It turned

out that distinction between neighboring cloud layers caused most confusion in the models presented here, which also applies for human observers. One reason for this lies in the visual similarities of *mid-layer* clouds to *high-* or *low-layer* clouds, as they contain a varying mix of water and ice particles. On the other hand, pure water- (*low-layer*) and ice-clouds (*high-layer*), which also have most significant optical differences, are very accurately distinguished from each other.

Despite the relevance of automated classification into different cloud layers, the pure detection of clouds is still of great importance for applications like ramp rate prediction. To evaluate binary segmentation, a binary model was retrained with the ground-truth containing only the classes *cloud* and (cloudless) *sky*. When comparing the results with postprocessed predictions of the 4-class CNN, it emerged that retraining improves accuracy about 0.33% points. Nonetheless, both versions of the model achieved excellent results with an average pixel-accuracy around 95%. Binary segmentation was also validated with a state-of-the art method. Compared to the accuracy achieved by a CSL on the same data, the CNN model prevails by a large margin of 7% points and for *mid-* and *high-layer* clouds even by more than 10% points.

An expressive comparison with the literature is hard to conduct due to discrepancies in the utilized data. Often, only sky patches are utilized [89] or difficult conditions like high-layer clouds [91] and overlapping multi-layer conditions [64] are omitted on purpose. In principle, image data from different observation sites need to be carefully selected using well-defined conditions to conduct comparable benchmarks [111].

Regarding semantic segmentation of multiple cloud types, there is only one other work using an expressive amount of data from 2019 [94]. There, the presented method beats a CNN on their dataset, but they do not apply self-supervised learning and use another architecture. However, considering the latest publications on cloud segmentation [86, 87, 88], a trend for applying deep learning in ground-based sky imaging is clearly identifiable.

As this work outlines the first approach of self-supervised (or unsupervised) learning with ASIs, its full potential has not been reached yet. Further studies are required, examining other approaches and other datasets.

For instance, motion has not been considered in the presented methods. However, to human observers, a clear distinction of overlapping clouds is sometimes only possible when looking at video data. Methods like described in [112] that use optical flow to learn from motion or in [113] applying a Siamese-triplet network to learn to track objects are two examples.

Furthermore, a concept called *noise contrastive estimation* to enhance self-supervised learning has gained a lot of interest in the computer vision community [114, 115]. The idea is not limited to a specific pretext task but suggests an adaption of the loss function to incorporate *contrastive losses* [116]. These measure similarities of sample pairs in representation space enabling the model to learn better features.

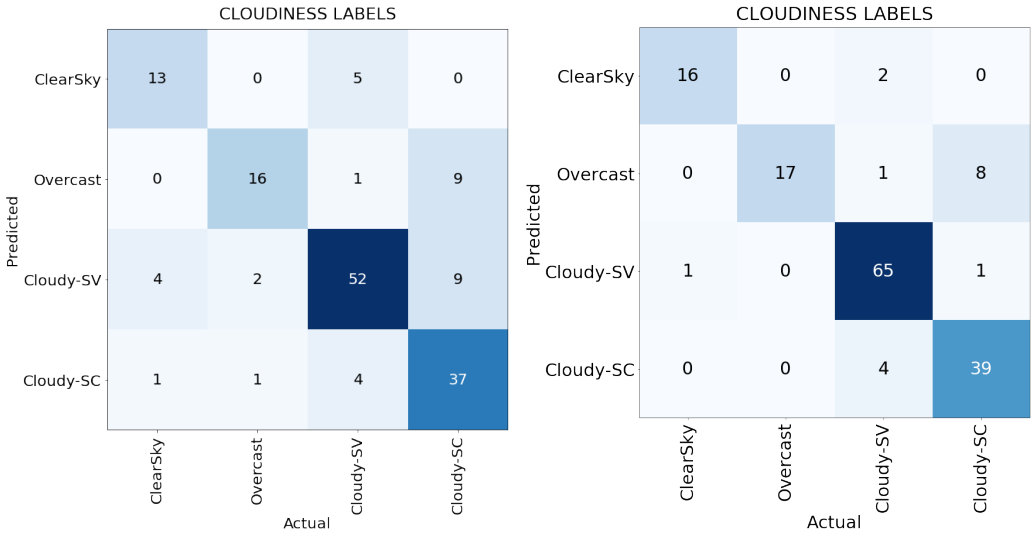
Apart from self-supervision, other techniques to learn representations could be useful, too. Semi-supervised learning which uses partially labeled and partially unlabeled images show also promising results in other vision applications [117]. Weakly-supervised learning could be applied by exploiting available data from other measurement instruments, like ceilometer data

to estimate height, or DNI measurements. Furthermore, instead of time-consuming labeling of single ASIs pixel-wise, video data could be annotated sequence-wise [118], resulting in much larger datasets for supervised pretraining.

To conclude, in this work the first approach of learning representations from ASIs with unlabeled data was presented. It was shown that pretraining and transfer learning in general are appropriate techniques to deal with limited annotated ASI data. As the presented approach outperformed a state-of-the-art method significantly, it is going to replace the CSL method currently applied in the DLR nowcasting system and also classification will be integrated. Regarding the amount of very recent publications dealing with representation learning, the gap between supervised and unsupervised methods seem to vanish. Therefore, I see a lot of potential in these techniques that should be further studied for ground-based cloud observations.

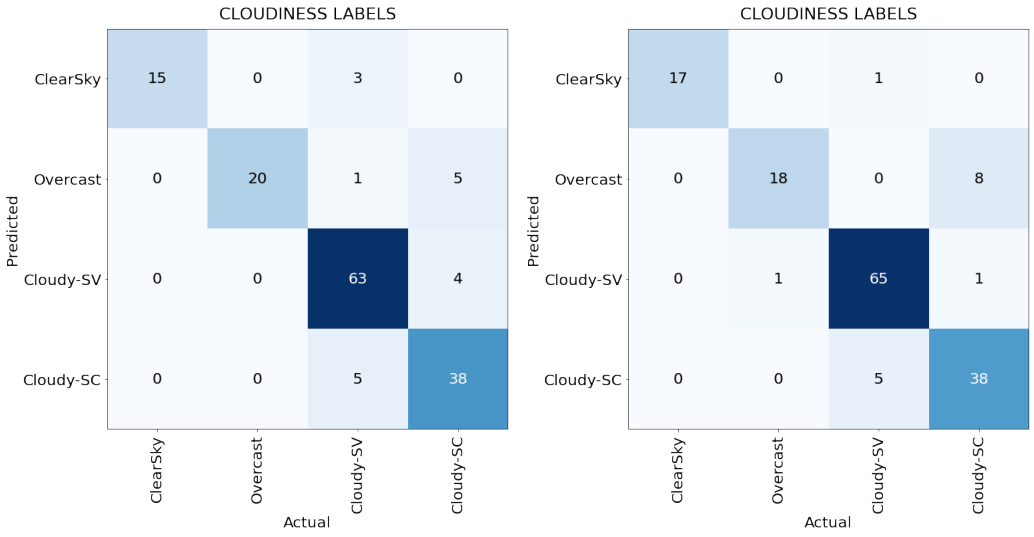
A. Detailed Comparison of Initializations

A.1. Multi-label Classification Results



(a) Random Init

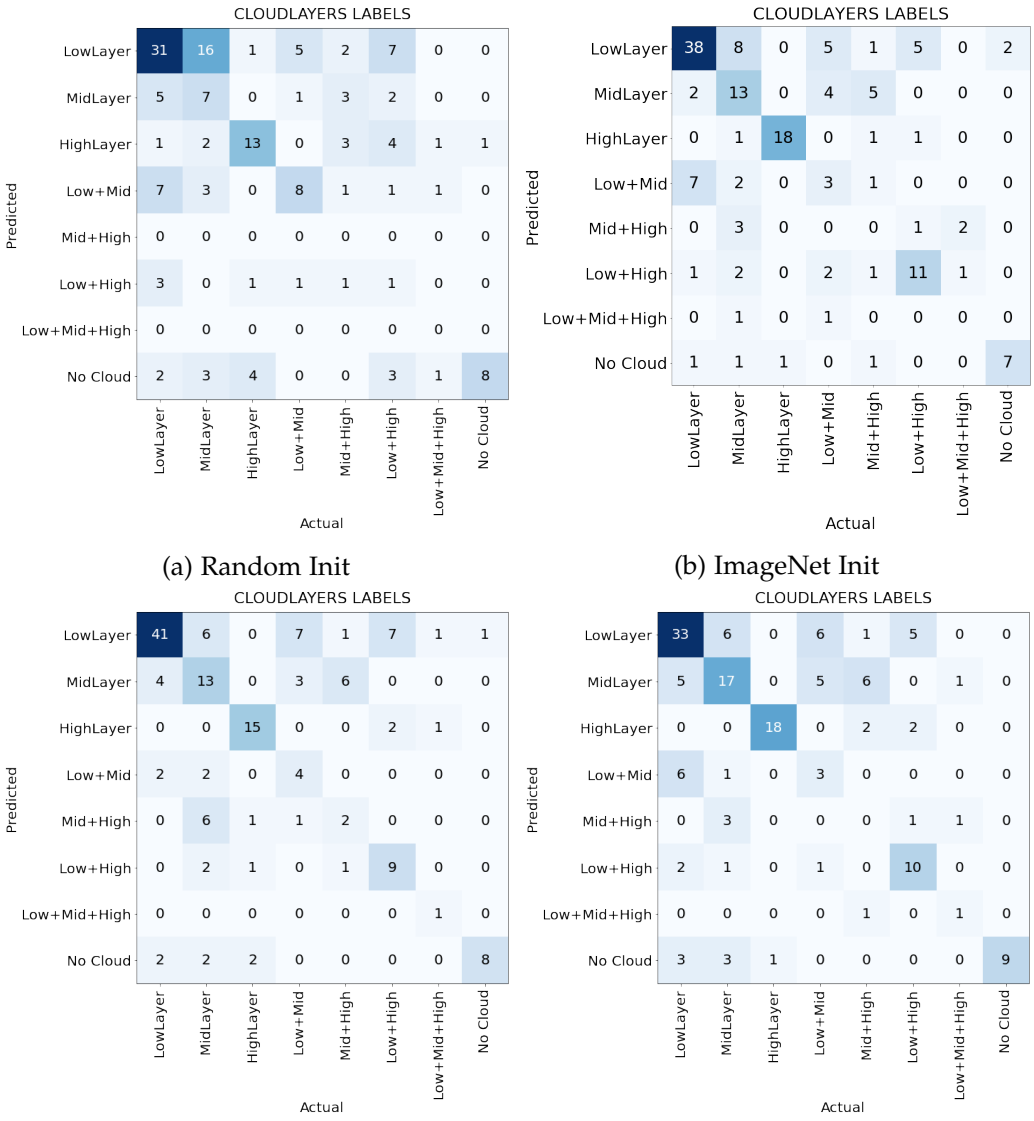
(b) ImageNet Init



(c) IP-SR Init

(d) DC Init

Figure A.1.: Confusion matrices for cloudiness label group



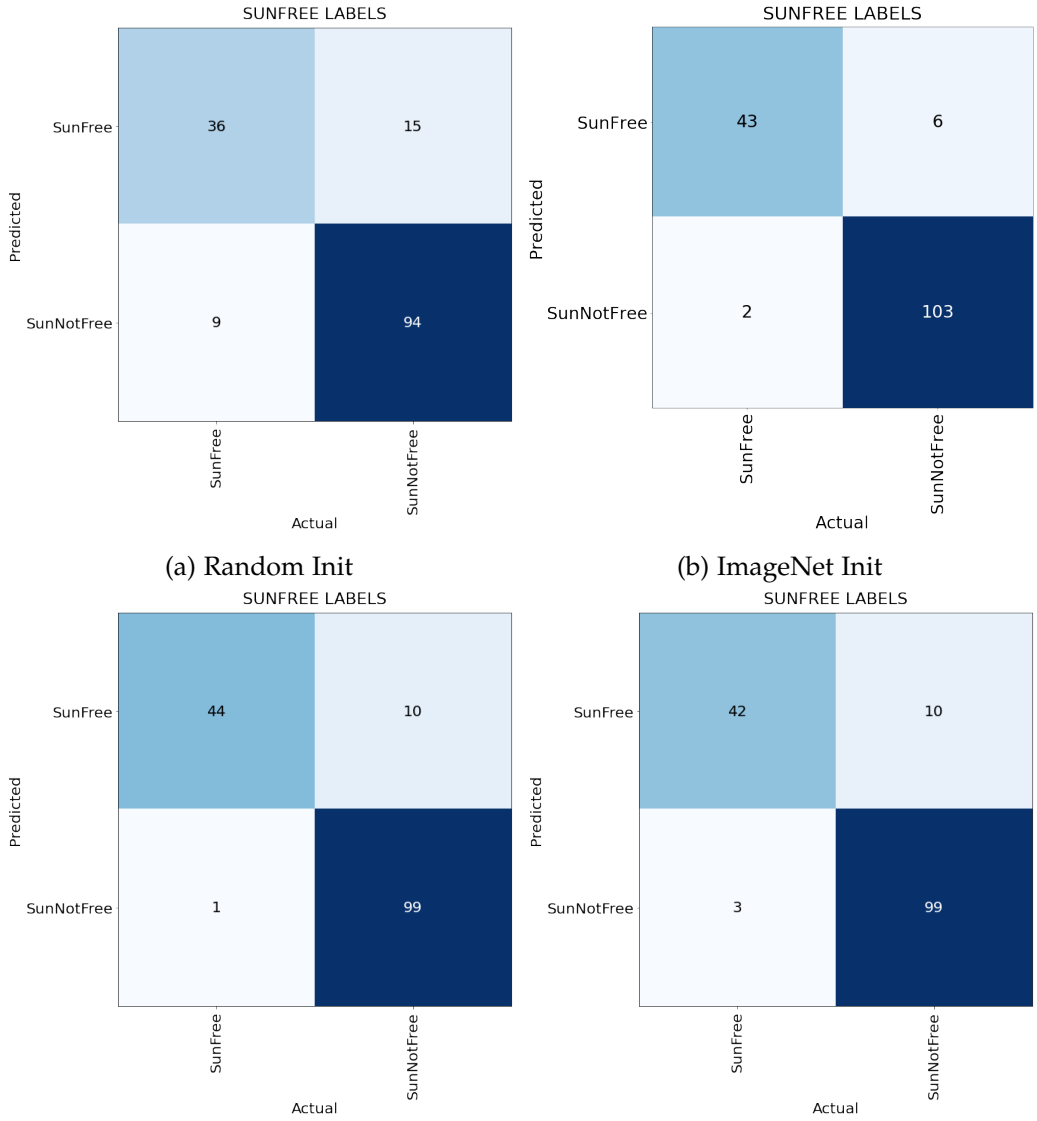
(a) Random Init

(b) ImageNet Init

(c) IP-SR Init

(d) DC Init

Figure A.2.: Confusion matrices for cloudlayer label group

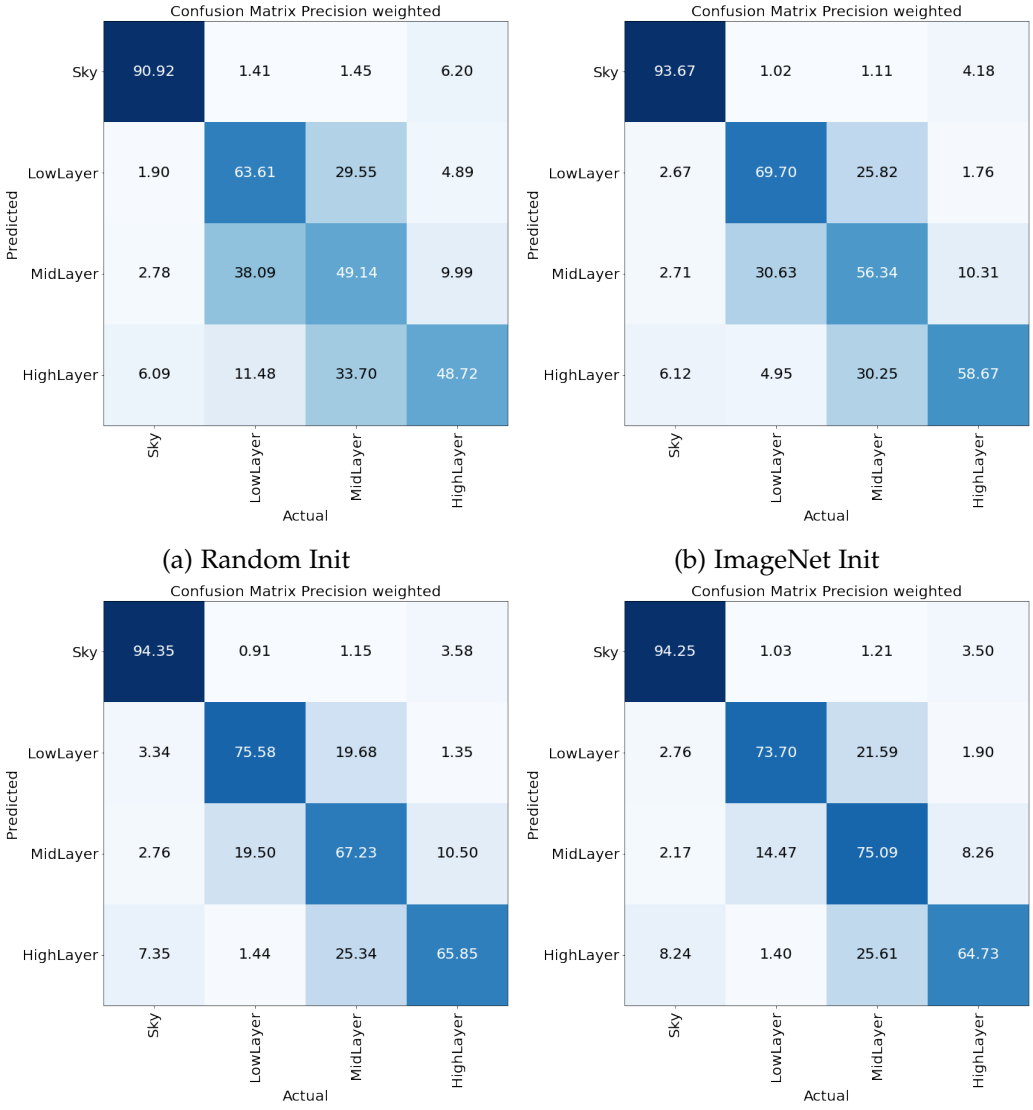


(a) Random Init (b) ImageNet Init

(c) IP-SR Init (d) DC Init

Figure A.3.: Confusion matrices for sun free label

A.2. Semantic Segmentation Results



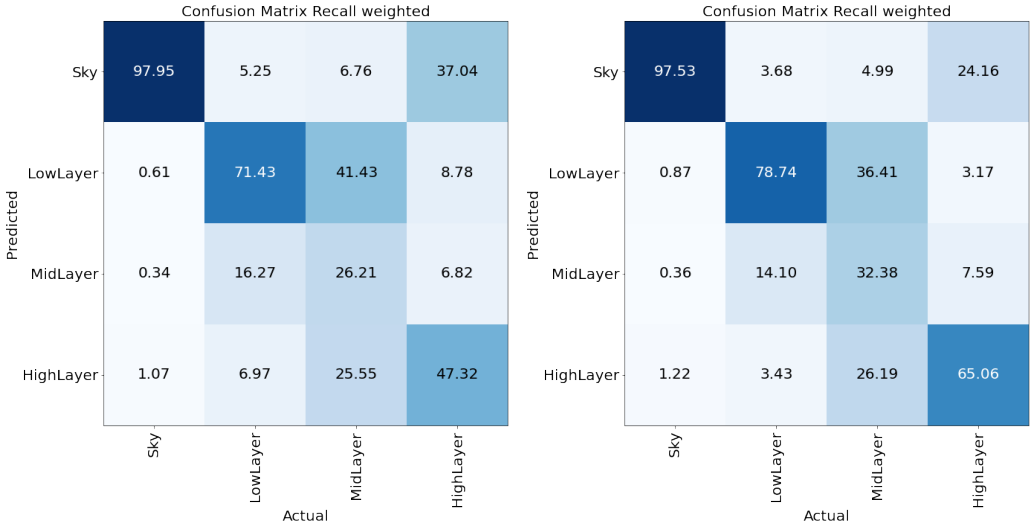
(a) Random Init

(b) ImageNet Init

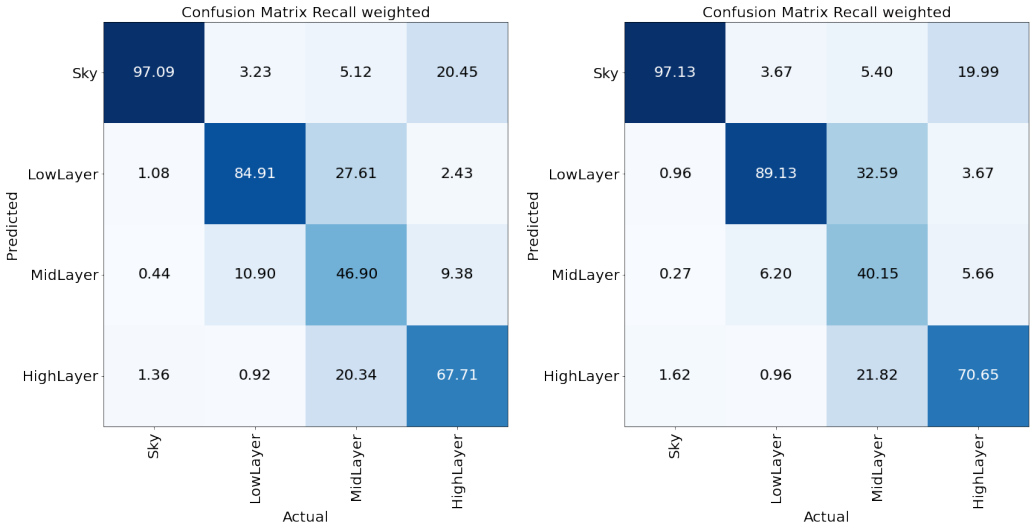
(c) IP-SR Init

(d) DC Init

Figure A.4.: Confusion matrices for precision of semantic segmentation

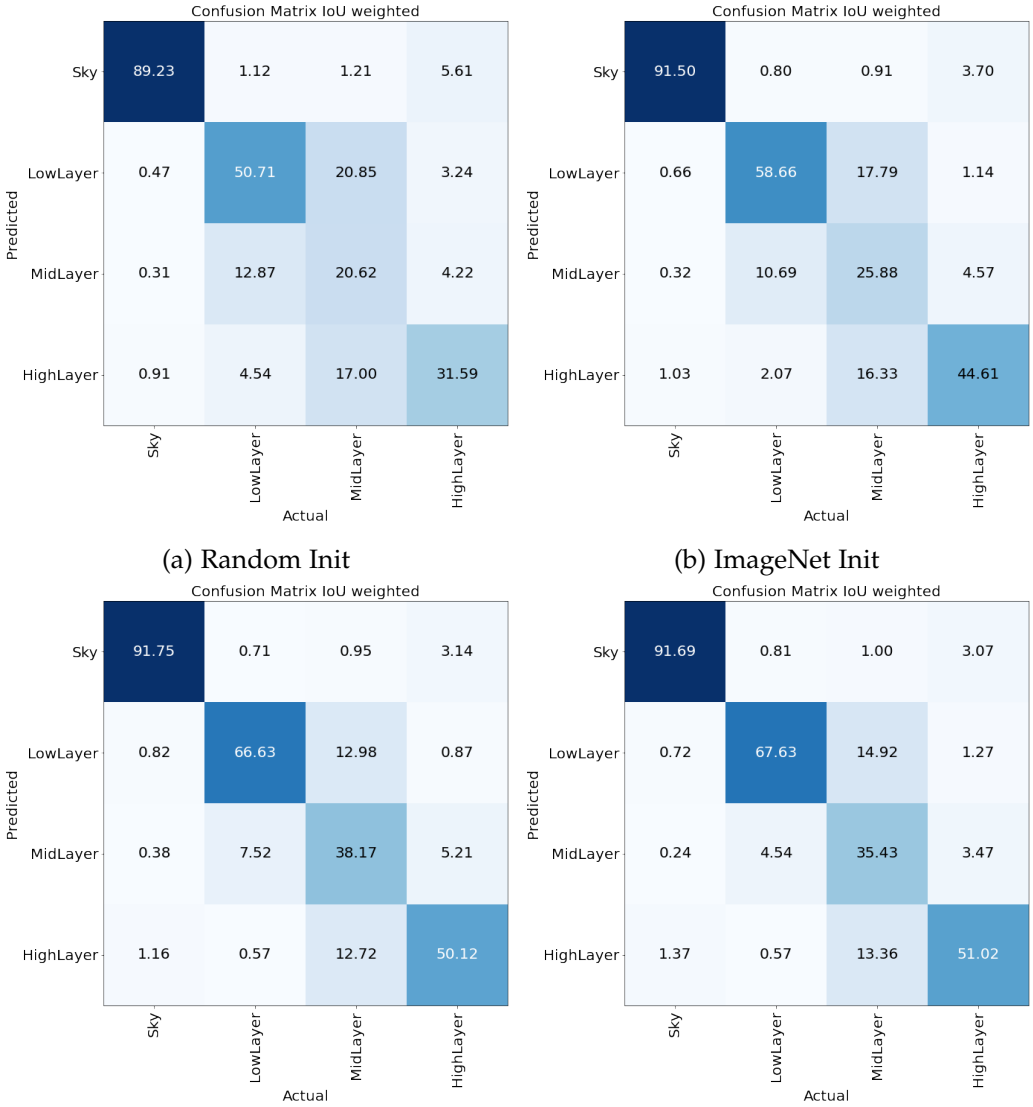


(a) Random Init (b) ImageNet Init



(c) IP-SR Init (d) DC Init

Figure A.5.: Confusion matrices for recall on semantic segmentation



(a) Random Init

(b) ImageNet Init

(c) IP-SR Init

(d) DC Init

Figure A.6.: Confusion matrices for IoU on semantic segmentation

List of Figures

List of Tables

Bibliography

[1] C. J. Hahn, W. B. Rossow, and S. G. Warren. "ISCCP cloud properties associated with standard cloud types identified in individual surface observations". In: *Journal of Climate* 14.1 (2001), pp. 11–28.

[2] M. Schroedter-Homscheidt, M. Kosmale, J. Sandra., and J. Kleissl. "Classifying ground-measured 1 minute temporal variability within hourly intervals for direct normal irradiances". In: *Meteorologische Zeitschrift* 27 (2018).

[3] B.-I. Crăciun, T. Kerekes, D. Séra, R. Teodorescu, and U. D. Annakkage. "Power ramp limitation capabilities of large PV power plants with active power reserves". In: *IEEE Transactions on Sustainable Energy* 8.2 (2016), pp. 573–581.

[4] P. Denholm and R. Margolis. *Energy Storage Requirements for Achieving 50% Solar Photovoltaic Energy Penetration in California*. Tech. rep. NREL/TP-6A20-66595. National Renewable Energy Laboratory, Aug. 2016.

[5] M. Saleh, L. Meek, M. A. Masoum, and M. Abshar. "Battery-less short-term smoothing of photovoltaic generation using sky camera". In: *IEEE Transactions on Industrial Informatics* 14.2 (2017), pp. 403–414.

[6] S. Watson, D. Bian, N. Sahraei, T. Buonassisi, I. M. Peters, et al. "Advantages of operation flexibility and load sizing for PV-powered system design". In: *Solar Energy* 162 (2018), pp. 132–139.

[7] I. L. García, J. L. Álvarez, and D. Blanco. "Performance model for parabolic trough solar thermal power plants with thermal storage: Comparison to operating plant data". In: *Solar Energy* 85.10 (2011), pp. 2443–2460.

[8] P. H. Wagner and M. Wittmann. "Influence of different operation strategies on transient solar thermal power plant simulation models with molten salt as heat transfer fluid". In: *Energy Procedia* 49 (2014), pp. 1652–1663.

[9] E. Lorenz, J. Remund, S. C. Müller, W. Traunmüller, G. Steinmaurer, D. Pozo, J. Ruiz-Arias, V. L. Fanego, L. Ramirez, M. G. Romeo, et al. "Benchmarking of different approaches to forecast solar irradiance". In: *24th European photovoltaic solar energy conference*. Hamburg Germany. 2009, pp. 21–25.

[10] M. Schroedter-Homscheidt and G. Gesell. "Verification of sectoral cloud motion based direct normal irradiance nowcasting from satellite imagery". In: *AIP Conference Proceedings*. Vol. 1734. 1. AIP Publishing LLC. 2016, p. 150007.

[11] P. Blanc, J. Remund, and L. Vallance. "Short-term solar power forecasting based on satellite images". In: *Renewable energy forecasting*. Elsevier, 2017, pp. 179–198.

[12] J. M. Bright, S. Killinger, D. Lingfors, and N. A. Engerer. "Improved satellite-derived PV power nowcasting using real-time power data from reference PV systems". In: *Solar Energy* 168 (2018), pp. 118–139.

[13] J. Shields, M. Karr, T. Tooman, D. Sowle, and S. Moore. "The whole sky imager–a year of progress". In: *Eighth Atmospheric Radiation Measurement (ARM) Science Team Meeting, Tucson, Arizona*. Citeseer. 1998, pp. 23–27.

[14] C. Long, D. Slater, and T. P. Tooman. *Total sky imager model 880 status and testing results*. Pacific Northwest National Laboratory Richland, Wash, USA, 2001.

[15] K. Widener and C. Long. *All sky imager*. US Patent App. 10/377,042. Sept. 2004.

[16] C. W. Chow, B. Urquhart, M. Lave, A. Dominguez, J. Kleissl, J. Shields, and B. Washom. "Intra-hour forecasting with a total sky imager at the UC San Diego solar energy testbed". In: *Solar Energy* 85.11 (2011), pp. 2881–2893.

[17] S. Quesada-Ruiz, Y. Chu, J. Tovar-Pescador, H. Pedro, and C. Coimbra. "Cloud-tracking methodology for intra-hour DNI forecasting". In: *Solar Energy* 102 (2014), pp. 267–275.

[18] A. Kazantzidis, P. Tzoumanikas, P. Blanc, P. Massip, S. Wilbert, and L. Ramirez-Santigosa. "Short-term forecasting based on all-sky cameras". In: *Renewable energy forecasting*. Elsevier, 2017, pp. 153–178.

[19] B. Nouri, P. Kuhn, S. Wilbert, C. Prahl, R. Pitz-Paal, P. Blanc, T. Schmidt, Z. Yasser, L. R. Santigosa, and D. Heineman. "Nowcasting of DNI maps for the solar field based on voxel carving and individual 3D cloud objects from all sky images". In: *AIP Conference Proceedings*. Vol. 2033. 1. AIP Publishing LLC. 2018, p. 190011.

[20] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. "A review on deep learning techniques applied to semantic segmentation". In: *arXiv preprint arXiv:1704.06857* (2017).

[21] J. Calbó, C. N. Long, J.-A. González, J. Augustine, and A. McComiskey. "The thin border between cloud and aerosol: Sensitivity of several ground based observation techniques". In: *Atmospheric Research* 196 (2017), pp. 248–260.

[22] I. Koren, L. A. Remer, Y. J. Kaufman, Y. Rudich, and J. V. Martins. "On the twilight zone between clouds and aerosols". In: *Geophysical Research Letters* 34.8 (2007).

[23] F. Linke. "Transmission coefficient and turbidity factor". In: *J Beitraege Phys Fr Atom* 10.2 (1922), pp. 91–103.

[24] Y. Zheng, S. Lin, C. Kambhamettu, J. Yu, and S. B. Kang. "Single-image vignetting correction". In: *IEEE transactions on pattern analysis and machine intelligence* 31.12 (2008), pp. 2243–2256.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[26] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[27] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. `www.deeplearningbook.org`. MIT Press, 2016.

[28] J. Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[29] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[31] A. Graves, A.-r. Mohamed, and G. Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

[32] M. D. Zeiler and R. Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[33] A. Saha. *Network In Network architecture: The beginning of Inception*. `http://teleported.in/posts/network-in-network/`. Accessed: 2020-07-02.

[34] I. Poletaev, K. Pervunin, M. Tokarev, et al. "Artificial neural network for bubbles pattern recognition on the images". In: *Journal of Physics: Conference Series*. Vol. 754. 7. IOP Publishing. 2016.

[35] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[36] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[37] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 1998, pp. 9–50.

[38] X. Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[39] D. Mishkin and J. Matas. "All you need is a good init". In: *arXiv preprint arXiv:1511.06422* (2015).

[40] K. He, X. Zhang, S. Ren, and J. Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[41] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[42] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[43] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[44] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[45] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[46] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[47] C. Shorten and T. M. Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of Big Data* 6.1 (2019), p. 60.

[48] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[49] K. He, R. Girshick, and P. Dollár. "Rethinking imagenet pre-training". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 4918–4927.

[50] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio. "Transfusion: Understanding transfer learning with applications to medical imaging". In: *arXiv preprint arXiv:1902.07208* (2019).

[51] V. R. de Sa. "Learning classification with unlabeled data". In: *Advances in neural information processing systems*. 1994, pp. 112–119.

[52] L. Jing and Y. Tian. "Self-supervised visual feature learning with deep neural networks: A survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[53] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec (2010), pp. 3371–3408.

[54] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. "Stacked convolutional auto-encoders for hierarchical feature extraction". In: *International conference on artificial neural networks*. Springer. 2011, pp. 52–59.

[55] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[56] A. Radford, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[57] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[58] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. "Photo-realistic single image super-resolution using a generative adversarial network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4681–4690.

[59] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European conference on computer vision*. Springer. 2016, pp. 694–711.

[60] C. Doersch, A. Gupta, and A. A. Efros. "Unsupervised visual representation learning by context prediction". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1422–1430.

[61] M. Noroozi and P. Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles". In: *European Conference on Computer Vision*. Springer. 2016, pp. 69–84.

[62] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. "Unsupervised representation learning by sorting sequences". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 667–676.

[63] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. "Deep clustering for unsupervised learning of visual features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 132–149.

[64] A. Heinle, A. Macke, and A. Srivastav. "Automatic cloud classification of whole sky images". In: *Atmospheric Measurement Techniques* 3.3 (2010), pp. 557–567.

[65] W. Zhuo, Z. Cao, and Y. Xiao. "Cloud classification of ground-based images using texture–structure features". In: *Journal of Atmospheric and Oceanic Technology* 31.1 (2014), pp. 79–92.

[66] L. Ye, Z. Cao, and Y. Xiao. "DeepCloud: Ground-based cloud image categorization using deep convolutional features". In: *IEEE Transactions on Geoscience and Remote Sensing* 55.10 (2017), pp. 5729–5740.

[67] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. "Image classification with the fisher vector: Theory and practice". In: *International journal of computer vision* 105.3 (2013), pp. 222–245.

[68] WMO. *International Cloud Atlas, Vol. I–Manual on the observations of clouds and other meteors*. 1975.

[69] J. Zhang, P. Liu, F. Zhang, and Q. Song. "CloudNet: Ground-based cloud classification with deep convolutional neural network". In: *Geophysical Research Letters* 45.16 (2018), pp. 8665–8672.

[70] C. N. Long, J. M. Sabburg, J. Calbó, and D. Pages. "Retrieving cloud characteristics from ground-based daytime color all-sky images". In: *Journal of Atmospheric and Oceanic Technology* 23.5 (2006), pp. 633–652.

[71] Q. Li, W. Lu, and J. Yang. "A hybrid thresholding algorithm for cloud detection on ground-based color images". In: *Journal of atmospheric and oceanic technology* 28.10 (2011), pp. 1286–1296.

[72] A. Kazantzidis, P. Tzoumanikas, A. F. Bais, S. Fotopoulos, and G. Economou. "Cloud detection and classification with the use of whole-sky ground-based images". In: *Atmospheric Research* 113 (2012), pp. 80–88.

[73] M. P. Souza-Echer, E. B. Pereira, L. Bins, and M. Andrade. "A simple method for the assessment of the cloud cover state in high-latitude regions by a ground-based digital camera". In: *Journal of Atmospheric and Oceanic Technology* 23.3 (2006), pp. 437–447.

[74] V. T. Jayadevan, J. J. Rodriguez, and A. D. Cronin. "A new contrast-enhancing feature for cloud detection in ground-based sky images". In: *Journal of Atmospheric and Oceanic Technology* 32.2 (2015), pp. 209–219.

[75] S. Liu, L. Zhang, Z. Zhang, C. Wang, and B. Xiao. "Automatic cloud detection for all-sky images using superpixel segmentation". In: *IEEE Geoscience and Remote Sensing Letters* 12.2 (2014), pp. 354–358.

[76] S. Liu, Z. Zhang, B. Xiao, and X. Cao. "Ground-based cloud detection using automatic graph cut". In: *IEEE Geoscience and remote sensing letters* 12.6 (2015), pp. 1342–1346.

[77] C. Shi, Y. Wang, C. Wang, and B. Xiao. "Ground-based cloud detection using graph model built upon superpixels". In: *IEEE Geoscience and Remote Sensing Letters* 14.5 (2017), pp. 719–723.

[78] M. Ghonima, B. Urquhart, C. Chow, J. Shields, A. Cazorla, and J. Kleissl. "A method for cloud detection and opacity classification based on ground based sky imagery". In: *Atmospheric Measurement Techniques Discussions* 5.4 (2012), pp. 4535–4569.

[79] R. Chauvin, J. Nou, S. Thil, A. Traore, and S. Grieu. "Cloud detection methodology based on a sky-imaging system". In: *Energy Procedia* 69 (2015), pp. 1970–1980.

[80] S. Wilbert, B. Nouri, C. Prahl, G. Garcia, L. Ramirez, L. Zarzalejo, R. Valenzuela, F. Ferrera, N. Kozonek, and J. Liria. "Application of whole sky imagers for data selection for radiometer calibration". In: *EU PVSEC 2016 Proceedings* (2016), pp. 1493–1498.

[81] P. Kuhn, B. Nouri, S. Wilbert, C. Prahl, N. Kozonek, T. Schmidt, Z. Yasser, L. Ramirez, L. Zarzalejo, A. Meyer, et al. "Validation of an all-sky imager–based nowcasting system for industrial PV plants". In: *Progress in Photovoltaics: Research and Applications* 26.8 (2018), pp. 608–621.

[82] S. Dev, Y. H. Lee, and S. Winkler. "Systematic study of color spaces and components for the segmentation of sky/cloud images". In: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2014, pp. 5102–5106.

[83] S. Dev, B. Wen, Y. H. Lee, and S. Winkler. "Machine learning techniques and applications for ground-based image analysis". In: *arXiv preprint arXiv:1606.02811* (2016).

[84] A. Taravat, F. Del Frate, C. Cornaro, and S. Vergari. "Neural networks and support vector machine algorithms for automatic cloud classification of whole-sky ground-based images". In: *IEEE Geoscience and remote sensing letters* 12.3 (2014), pp. 666–670.

[85] H.-Y. Cheng and C. L. Lin. "Cloud detection in all-sky images via multi-scale neighborhood features and multiple supervised learning techniques." In: *Atmospheric Measurement Techniques* 10.1 (2017).

[86] S. Dev, S. Manandhar, Y. H. Lee, and S. Winkler. "Multi-label cloud segmentation using a deep network". In: *2019 USNC-URSI Radio Science Meeting (Joint with AP-S Symposium)*. IEEE. 2019, pp. 113–114.

[87] W. Xie, D. Liu, M. Yang, S. Chen, B. Wang, Z. Wang, Y. Xia, Y. Liu, Y. Wang, and C. Zhang. "SegCloud: a novel cloud image segmentation model using a deep convolutional neural network for ground-based all-sky-view camera observation". In: *Atmospheric Measurment Techniques* 13 (2020), pp. 1953–1961.

[88] Q. Song, Z. Cui, and P. Liu. "An Efficient Solution for Semantic Segmentation of Three Ground-based Cloud Datasets". In: *Earth and Space Science* 7.4 (2020).

[89] S. Dev, Y. H. Lee, and S. Winkler. "Color-based segmentation of sky/cloud images from ground-based cameras". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10.1 (2016), pp. 231–242.

[90] S. Dev, Y. H. Lee, and S. Winkler. "Multi-level semantic labeling of sky/cloud images". In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2015, pp. 636–640.

[91] M. Hasenbalg, P. Kuhn, S. Wilbert, B. Nouri, and A. Kazantzidis. "Benchmarking of six cloud segmentation algorithms for ground-based all-sky imagers". In: *Solar Energy* 201 (2020), pp. 596–614.

[92] R. Adams and L. Bischof. "Seeded region growing". In: *IEEE Transactions on pattern analysis and machine intelligence* 16.6 (1994), pp. 641–647.

[93] B. Nouri, S. Wilbert, L. Segura, P. Kuhn, N. Hanrieder, A. Kazantzidis, T. Schmidt, L. Zarzalejo, P. Blanc, and R. Pitz-Paal. "Determination of cloud transmittance for all sky imager based solar nowcasting". In: *Solar Energy* 181 (2019), pp. 251–263.

[94] L. Ye, Z. Cao, Y. Xiao, and Z. Yang. "Supervised Fine-Grained Cloud Detection and Recognition in Whole-Sky Images". In: *IEEE Transactions on Geoscience and Remote Sensing* 57.10 (2019), pp. 7972–7985.

[95] *WMO Definition of Clouds*. `https://cloudatlas.wmo.int/en/clouds-definitions.html`. Accessed: 2020-06-07.

[96] K. Sassen and Z. Wang. "The clouds of the middle troposphere: Composition, radiative impact, and global distribution". In: *Surveys in geophysics* 33.3-4 (2012), pp. 677–691.

[97] S. Dev, Y. H. Lee, and S. Winkler. "Categorization of cloud image patches using an improved texton-based approach". In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2015, pp. 422–426.

[98] B. Nouri, S. Wilbert, P. Kuhn, N. Hanrieder, M. Schroedter-Homscheidt, A. Kazantzidis, L. Zarzalejo, P. Blanc, S. Kumar, N. Goswami, et al. "Real-Time Uncertainty Specification of All Sky Imager Derived Irradiance Nowcasts". In: *Remote Sensing* 11.9 (2019), p. 1059.

[99] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. "On label dependence and loss minimization in multi-label classification". In: *Machine Learning* 88.1-2 (2012), pp. 5–45.

[100] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1874–1883.

[101] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi. "Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize". In: *arXiv preprint arXiv:1707.02937* (2017).

[102] J. Howard and S. Gugger. "Fastai: A layered API for deep learning". In: *Information* 11.2 (2020), p. 108.

[103] A. Krizhevsky, V. Nair, and G. Hinton. *The CIFAR-10 dataset*. `https://www.cs.toronto.edu/~kriz/cifar.html`. Accessed: 2020-07-03.

[104] S. Qianqian. *Replication Data for: An Efficient Solution for Semantic Segmentation of Three Ground-based Cloud datasets*. Publication Date: 2020-03-13. DOI: `\url{https://doi.org/10.7910/DVN/HEJTK1}`.

[105] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[106] L. N. Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 464–472.

[107] L. N. Smith. "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay". In: *preprint arXiv:1803.09820* (2018).

[108] I. Loshchilov and F. Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[109] facebook research. *faiss*. `https://github.com/facebookresearch/faiss`. 2020.

[110] J. Johnson, M. Douze, and H. Jégou. "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* (2019).

[111] B. Nouri, S. Wilbert, N. Blum, P. Kuhn, T. Schmidt, Z. Yasser, L. Zarzalejo, F. Lopes, H. Silva, M. Schroedter-Homscheidt, A. Kazantzidis, C. Raeder, P. Blanc, and R. Pitz-Paal. "Evaluation of an All Sky Imager Based Nowcasting System for Distinct Conditions and Five Sites. 25th SolarPACES Conference". 2019.

[112] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. "Learning features by watching objects move". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2701–2710.

[113] X. Wang and A. Gupta. "Unsupervised learning of visual representations using videos". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2794–2802.

[114] I. Misra and L. v. d. Maaten. "Self-supervised learning of pretext-invariant representations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6707–6717.

[115] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. "Momentum contrast for unsupervised visual representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738.

[116] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[117] P. Haeusser, A. Mordvintsev, and D. Cremers. "Learning by Association–A Versatile Semi-Supervised Training Method for Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 89–98.

[118] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon. "Multiple instance learning: A survey of problem characteristics and applications". In: *Pattern Recognition* 77 (2018), pp. 329–353.