LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME
UND DATA MINING

Master Thesis

Unsupervised Semantic Segmentation of Coarse Woody Debris in Aerial Images

Maximilian Bernhard

Supervisor: Prof. Dr. Matthias Schubert
Submission date: 28.10.2020

Abstract

Coarse woody debris, i.e. fallen trees and larger branches, plays an important role in the ecosystem of forests. Hence, a pixelwise segmentation of coarse woody debris in aerial images can be a valuable source of information for forestal management. At the same time, unsupervised machine learning has the advantage that it does not require the costly acquisition of ground-truth annotations for training.

In this master thesis, we assess several existing methods for the task of unsupervised semantic segmentation of coarse woody debris. We also propose a novel two-step procedure that separates the extraction of segments and the assignment of semantic labels to them. So, the first step consists of a segmentation algorithm, acting as an extractor of candidate segments. In the second step, we employ state-of-the-art image clustering methods for the label assignment.

Our experiments show that this approach significantly outperforms the existing methods for unsupervised semantic segmentation which fail to properly detect the small and hardly obvious features of coarse woody debris.

Contents

1 Introduction

Over the past 50 years, the boreal forests of Alberta, Canada, have suffered lots of changes caused by humans. In particular, thousands of so-called *seismic lines* were created for the purpose of oil exploration. These are corridors in the forests where trees and other obstacles have been removed in order to facilitate the use of heavy machinery in this terrain. They are called seismic lines because they were built for mapping underground oil occurrences with seismic sound waves [42]. Although the area of destroyed forest is comparably small, the existence of seismic lines poses a major change for wildlife in the region. For instance, woodland caribou populations declined dramatically as a consequence. The reason for that is that seismic lines provide space and food for moose and other deer which originally do not occur in this habitat. However, with the seismic lines, moose began to populate these forests which in turn also brought more wolves to this area. Ultimately, wolf predation of caribou increased, leading to a decline in population [14].
Therefore, it is desirable to reforest seismic lines and to restore the natural habitat. Unfortunately, the forests recover very slowly on seismic lines which is why researchers suppose that human restoration treatments may be beneficial [14]. But, current methods of reforestation are expensive.
Hence, it is expedient to apply such treatments in a purposeful and targeted manner. *Coarse woody debris*, being logs, snags and large branches of dead trees, can play an important role in this regard. It provides valuable nutrients for seedlings, but it also has some other valuable characteristics. So, knowledge about the amount and the location of coarse woody debris can be used to make forest management and restoration more effective [28].
This knowledge can be extracted from aerial images, which are nowadays relatively easy to obtain. Unfortunately, the amounts of data usually tend to be extremely large making a manual analysis cumbersome. An automated analysis of such data, on the contrary, can save a lot of resources and money.
In this work, we want to develop a method for the detection of coarse woody debris using machine learning. More concretely, we aim to find a semantic segmentation model that is able to localize the logs on pixel level. The major benefit of semantic image segmentation is that predictions are made on pixel level, which is the most precise level possible for images. Other common methods like object detection with bounding boxes are not as precise. This holds especially for the case of coarse woody debris. For example, a tree lying diagonally in a rectangular bounding box covers only a small area of the box while the majority of the area can be seen as “dead space”. This property may not be overly problematic for the use case at hand, but in general, there are situations, e.g. autonomous driving, where semantic segmentation may be the

clearly better choice.
Furthermore, we approach this problem with unsupervised learning. That is, during training, the models do not receive any supervisory signals indicating the true class labels of certain image regions. Unsupervised machine learning has the big advantage that it is not necessary to create ground-truth annotations for the training data, which can not only be very costly but also extremely time-consuming or even impossible in some situations. Particularly for segmentation, where the ground truth has to contain a label for every pixel, the acquisition of labels is expensive. On the other hand, unsupervised learning on images is quite challenging as this kind of data has a very high dimensionality and, thus, exhibits lots of low-level features. This makes it difficult to filter out the important features when no supervision is available. Consequently, existing approaches for unsupervised image segmentation have some limitations and there is comparably few literature in this field [16].
Subsequently, we will shortly present a selection of studies on the detection of coarse woody debris. Then, we will discuss some methods we consider the state of the art in unsupervised image segmentation in more detail. We will also propose a - to the best of our knowledge - novel framework for generating image segmentations in an unsupervised way, which mitigates some of the shortcomings of the existing methods. Thereafter, we will describe the dataset that was used for this study and compare the discussed approaches in an empirical analysis before we draw our final conclusions.

2 Related Work on the Detection of Coarse Woody Debris

In this section, we will provide a brief overview of existing work on the detection of coarse woody debris and windthrown trees. We will focus on the differences between these works and our problem setting.

2.1 Mapping Coarse Woody Debris with Random Forest Classification of Centimetric Aerial Imagery

The problem of mapping coarse woody debris in aerial images of boreal forests has been addressed in [28]. In fact, the study area and the imagery in their work are identical to ours.
The authors employed a GEOBIA (Geographic Object-based Image Analysis) [17] approach with random forest classifiers to create semantic segmentations of the aerial images. In particular, a not further described segmentation algorithm was used to divide the image into regions that were represented as image objects with some attributes. These included, for example, the means and standard deviations of the image channels, spatial attributes and also information about the neighboring segments. Then, these attributed image objects were classified with a random forest in a supervised manner.
The predicted classes were *Log*, *Snag*, *Water*, *Dirt* and *Other*. As input channels not only RGB values were available, but also near-infrared values, a *Canopy Height Model*, a *Digital Surface Model* and a so-called *Normalized Difference Vegetation Index*. A detailed explanation of these terms can be found in Section 5.1. With these data, their random forest model achieved a completeness of 80.6% and a correctness of 92.3% for the overall coarse woody debris class, being the union of *Log* and *Snag*.
So, our work and [28] are quite similar and [28] serves as a useful orientation when it comes to evaluation. However, there also exist some major differences. First of all, we aim to solve this problem without supervision. This makes our work from the viewpoint of the machine learning community conceptually very different and adds new value to it. Another conceptual difference is that, apart from the segmentation algorithm for extracting the image objects, no methods that are specific for computer vision were employed, while this is clearly our goal.
Aside from that, we focus only on the class *Coarse Woody Debris*, i.e. we neither distinguish between *Log* and *Snag* nor between *Water*, *Dirt* and *Other*. Technically, this is not a big reduction of complexity as every multi-class problem can be broken down into several binary problems, where every binary

problem handles one class.
There is also one issue in [28] that is not negligible in a comparison with our work. The authors created their ground-truth annotations for evaluation based on the same segmentation that was used for extracting the image objects for training. That is, the images were partitioned into segments and, then, those were labeled as a whole instead of providing pixelwise labels. This procedure relies on the assumption that the borders of the true segments are perfectly aligned with the segments produced by the segmentation algorithm. If not, i.e. if some segments cover more than two classes at the same time, this induces an error in the evaluation scores as the labeled segments do not perfectly represent the ground truth. Especially for the case of coarse woody debris, this may be problematic as the segments containing dead trees have a relatively large border compared to their total covered area. Thus, the borders of the labeled segments being perfectly aligned with the borders of the extracted image objects is a major advantage at test time. Ultimately, this can lead to an overestimation of the model performance. Nevertheless, their method produces high-quality segmentations for which we depict an example in Figure 1.

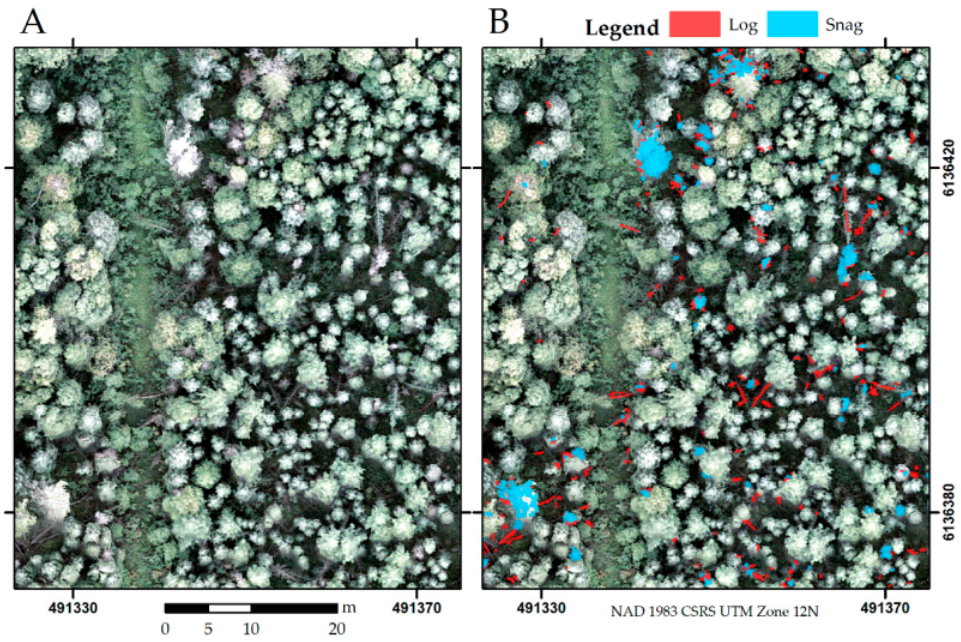


Figure 1: Example for the detection of logs and snags with the method of [28]. Logs (red) are identified very well, whereas the prediction seems to be less precise for snags (blue). *Source:* [28].

2.2 Coarse-to-Fine Windthrown Tree Extraction Based on Unmanned Aerial Vehicle Images

In [11], the authors developed an algorithm for the localization of windthrown trees. Their approach combines a binary random forest, a skeletonization algorithm and the Hough-transform [19]. The random forest was trained in a supervised way to roughly localize the windthrown trees. It was provided with features such as RGB values and other texture statistics like mean, variance and contrast. Then, these coarse objects were turned into skeletal lines to reduce the influence of the trunk diameter. Finally, windthrown trees were extracted as straight lines by means of the Hough-transform.
This method led to a completeness of 75.7 % and a correctness of 92.5 %. However, the evaluation was not performed on pixel level but on object level. That is, the authors counted the number of correctly and falsely detected trees instead of pixels. Thus, it is hard to draw conclusions on how good this method works for semantic segmentation of windthrown trees and the reported evaluation scores are rather irrelevant for our setting.
The study area is also quite different from ours. It contains imagery of a Chinese rubber tree plantation and totals an area of only 0.25ha. In contrast to that, our study area is much larger, more natural and, therefore, more disturbed and diverse. This does also not allow for a direct, quantitative comparison of the two works.

2.3 Detection of Fallen Logs from High-resolution UAV Images

A similar approach to the previous was proposed in [34]. Here the authors developed an algorithmic method for the detection of fallen logs. They employ a combination of filterings for edge detection in a preprocessing phase and the Hough-transform for the final identification of the logs. That means, this is a purely handcrafted algorithm and there is no machine learning involved. Despite that, it is able to perform well in their experiments, achieving an accuracy of 94.9 %. Once again, the evaluation was conducted on object level, so the scores can not be transferred to image segmentation.
The study area is located in West Bohemia and consists only of six aerial images, each covering an area of 50m×50m. On top of that, these patches exhibit an open canopy. Therefore, the fallen logs are well visible in almost all of the cases. As we explained in the previous section, our large and diverse study area is fundamentally different, which is why one can not expect the results of [34] to generalize well on our data. Nevertheless, it shows that relatively simple methods suffice to capture the characteristics of coarse woody debris

in aerial imagery. This might be useful for building a strong preprocessing pipeline that alleviates learning for our models.

3 Related Work in the Field of Unsupervised Image Segmentation

Next, we will discuss some important approaches in the field of unsupervised image segmentation. To start with, we want to give a short overview in the form of a categorization of recent methods.
Approaches like [22, 33] aim to maximize the mutual information between different versions of an image. Therefore, these **MI-based methods** also have a flavor of contrastive learning as pairs of images are used to train a network. A major benefit of these methods is that they allow achieving an invariance under some augmentations or transformations that do not semantically alter the image contents.
Iterative methods, e.g. [25, 32], employ a training scheme where initial hidden representations and segmentations are updated multiple times by back-propagating a loss that enforces certain constraints like consistency or spatial continuity. For instance, in [32], clustering assignments serve as pseudo-labels, while [25] utilizes generated labels of superpixels to train their model.
The W-Net [45] can be seen as an **encoder-decoder architecture**. Here, the encoded representation is a segmentation mask which is then used to reconstruct the original image again. By applying loss functions on both the encoded and the decoded states, the produced segmentation mask is forced to have preferable properties.
There are also approaches that rely on **generative models**, e.g. [2, 24]. In these approaches, a generator creates, among side products, a segmentation that is desirably as realistic as possible and, then, assessed by a discriminator network. Throughout the training, the discriminator learns to better distinguish between real and generated segmentations making the generator produce more and more plausible segmentations.
Aside from the mentioned frameworks, methods for **unsupervised representation learning** are often adopted for segmentation in the literature. For instance, [22, 33] apply K-Means clustering on pixel representations learned with the methods [5, 10, 20] and use these as baselines for their models.
Subsequently, we will thoroughly describe some methods which we use as baselines and consider the state of the art for our problem. On top of that, we will also give a brief introduction to other approaches that seemed less suitable for our setting.

3.1 Invariant Information Clustering (IIC)

Invariant information clustering [22], or short IIC, is a technique that falls under the categories of MI-based techniques and contrastive learning. The core

idea is to randomly transform or perturb an image and then train a convolutional neural network, subsequently dubbed CNN, to maximize the mutual information between the outputs for the original and the augmented version. Applicable transformations include geometric transformations, like rotation or flipping, and photometric ones, like color jittering. The only requirement in this regard is that the information of interest, i.e. the true cluster of the image, is invariant under the perturbations. This framework can be employed for both image clustering and image segmentation. In the following, we will first describe the method in detail before we discuss its capabilities.

3.1.1 Learning Objective for Clustering

Formulation If we deal with images $\mathbf{x} \in \mathcal{D}$, it is desirable to learn representations $\Phi(\mathbf{x})$ that preserve the features of interest, e.g. the depicted object, and discard instance-specific details, e.g. spatial orientation. The latter property is crucial as a representation being the trivial identity function $\Phi(\mathbf{x}) = \mathbf{x}$ is clearly useless, although it preserves all necessary information about the image. Therefore, it is sensible to introduce a bottleneck to the representation function. In the case of image clustering, this is achieved by setting the output format to a categorical distribution. That is, $\Phi(\mathbf{x})$ is a C-dimensional vector produced by a Softmax layer, which indicates the probabilities that an image $\mathbf{x}$ belongs to the clusters $c \in \{1, ..., C\}$. Formally, this can be denoted as $\Phi_c(\mathbf{x}) = P(z = c \mid \mathbf{x})$, where z is a discrete random variable representing the true cluster assignment.
Now, assume we have pairs of samples $(\mathbf{x}, \tilde{\mathbf{x}})$ from a joint distribution $P(\mathbf{x}, \tilde{\mathbf{x}})$, for instance, two images containing the same object class. As we already mentioned, our objective is to maximize the mutual information of the representations, i.e.

$$\max_{\Phi} I(\Phi(\mathbf{x}), \Phi(\tilde{\mathbf{x}})),$$

while we maintain the clustering bottleneck. If we again denote the cluster assignment variables of $\mathbf{x}$ and $\tilde{\mathbf{x}}$ as z and $\tilde{z}$, their conditional joint distribution can be written as

$$P(z = c, \tilde{z} = \tilde{c} \mid \mathbf{x}, \tilde{\mathbf{x}}) = \Phi_c(\mathbf{x}) \cdot \Phi_{\tilde{c}}(\tilde{\mathbf{x}}). \tag{1}$$

That is, given two specific instances $\mathbf{x}$ and $\tilde{\mathbf{x}}$, z and $\tilde{z}$ are independent, but this does in general not hold for their marginal distribution over the whole dataset of paired samples. To clarify the explanation of [22], we provide an additional concrete example. Consider a dataset of only two pairs $(\mathbf{x}_i, \tilde{\mathbf{x}}_i)$ with $i = 1, 2$, where both images for $i = 1$ show cats and the images for $i = 2$ show dogs. Then, the cluster assignment variables take the values $z_1 =$ "*cat*", $\tilde{z}_1 =$ "*cat*",

$z_2 =$ "*dog*" and $\tilde{z}_2 =$ "*dog*". This results in

$$P(z = \text{“}cat\text{”}) = P(z = \text{“}dog\text{”}) = P(\tilde{z} = \text{“}cat\text{”}) = P(\tilde{z} = \text{“}dog\text{”}) = 0.5,$$

but

$$P(z = \text{“}cat\text{”}, \tilde{z} = \text{“}cat\text{”}) = P(z = \text{“}dog\text{”}, \tilde{z} = \text{“}dog\text{”}) = 0.5 \neq 0.5 \cdot 0.5.$$

This shows that z and $\tilde{z}$ are not independent; in fact, they are predictive for each other. In contrast to that, it is very reasonable that the cluster assignment z of a specific image $\mathbf{x}$ does not depend on the the assignment $\tilde{z}$ of another image $\tilde{\mathbf{x}}$ given the two images, which justifies the conditional independence of z and $\tilde{z}$ stated above.
When we marginalize the joint distribution of z and $\tilde{z}$ over the dataset or a batch, we obtain a $C \times C$ matrix $\mathbf{P}$ with $\mathbf{P}_{c\tilde{c}} = P(z = c, \tilde{z} = \tilde{c})$ for $c, \tilde{c} \in \{1, ..., C\}$. According to equation (1), we can compute $\mathbf{P}$ as

$$\begin{aligned}
\mathbf{P} &= \Big(P(z = c, \tilde{z} = \tilde{c})\Big)_{c\tilde{c}} \\
&= \left(\frac{1}{n}\sum_{i=1}^{n} P(z = c, \tilde{z} = \tilde{c} \mid \mathbf{x}_i, \tilde{\mathbf{x}}_i)\right)_{c\tilde{c}} \\
&= \left(\frac{1}{n}\sum_{i=1}^{n} \Phi_c(\mathbf{x}_i) \cdot \Phi_{\tilde{c}}(\tilde{\mathbf{x}}_i)\right)_{c\tilde{c}} \\
&= \frac{1}{n}\sum_{i=1}^{n} \Phi(\mathbf{x}_i) \cdot \Phi(\tilde{\mathbf{x}}_i)^\top.
\end{aligned}$$

With $\mathbf{P}$, we also get the marginals

$$\mathbf{P}_c = P(z = c) = \sum_{\tilde{c}=1}^{C} P(z = c, \tilde{z} = \tilde{c}) = \sum_{\tilde{c}=1}^{C} \mathbf{P}_{c\tilde{c}}$$

and

$$\mathbf{P}_{\tilde{c}} = P(\tilde{z} = \tilde{c}) = \sum_{c=1}^{C} P(z = c, \tilde{z} = \tilde{c}) = \sum_{c=1}^{C} \mathbf{P}_{c\tilde{c}}.$$

Furthermore, the authors of [22] propose to symmetrize $\mathbf{P}$ using $\frac{\mathbf{P}+\mathbf{P}^\top}{2}$ because swapping $\mathbf{x}$ and $\tilde{\mathbf{x}}$ as well as z and $\tilde{z}$ should not make any difference.
Finally, the mutual information of z and $\tilde{z}$ or $\Phi(\mathbf{x})$ and $\Phi(\tilde{\mathbf{x}})$ respectively can

be calculated as

$$\begin{aligned} I(z, \tilde{z}) &= I(\mathbf{P}) \\ &= \sum_{c=1}^{C} \sum_{\tilde{c}=1}^{C} P(z = c, \tilde{z} = \tilde{c}) \cdot \log \frac{P(z = c, \tilde{z} = \tilde{c})}{P(z = c) \cdot P(\tilde{z} = \tilde{c})} \\ &= \sum_{c=1}^{C} \sum_{\tilde{c}=1}^{C} \mathbf{P}_{c\tilde{c}} \cdot \log \frac{\mathbf{P}_{c\tilde{c}}}{\mathbf{P}_c \cdot \mathbf{P}_{\tilde{c}}} \end{aligned} \tag{2}$$

Degeneracy One huge benefit of this training objective is that it naturally avoids degenerate solutions for Φ. That is, maximizing this objective will neither lead to a Φ that predicts the same cluster for every image nor will it propose a uniform distribution over the clusters for all of the images.
Especially the first property is rather remarkable since we can only generate pairs of images belonging to the same cluster. The reason for that is that, in the fully unsupervised setting, we do not have any ground-truth labels. So, we can only use image transformations that do not have any impact on the true cluster assignment. Hence, during training, the model sees only pairs of images, that are both in the same cluster, and is therefore prone to predicting the same label for every image.
However, the mutual information as stated in equation (2) manages to avoid this, which can be easily understood with relations to other quantities in information theory. In [22], the relation $I(z, \tilde{z}) = H(z) - H(z \,|\, \tilde{z})$ is employed for the clarification of this property, but here, we utilize the similar but slightly different relation $I(z, \tilde{z}) = H(z) + H(\tilde{z}) - H(z, \tilde{z})$. It states that the mutual information is high when the entropies of both z and $\tilde{z}$ are high and simultaneously, their joint entropy is low.
For the extreme case that every image is assigned to the same cluster with a confidence of 100%, $H(z)$ and $H(\tilde{z})$ are both equal to zero, leading to $I(z, \tilde{z}) = 0$ (note that the mutual information is always nonnegative). Contrariwise, if a noninformative, uniform distribution over the clusters is predicted for every image, then $H(z)$ and $H(\tilde{z})$ take their maximum value being $\log C$. Fortunately, the joint entropy $H(z, \tilde{z})$ takes its maximum value $\log C^2 = 2 \cdot \log C$ in that case, too. Thus, the marginal entropies and the joint entropy cancel each other out, leading again to a mutual information of zero.
Altogether, this shows that the two demonstrated degenerate cases result in a mutual information of zero, the worst possible value. So, by training with this objective, we will surely end up with a function Φ that lies somewhere between those two extremes. More loosely spoken, all this means that a CNN trained with the mutual information objective tries to make the cluster assignment distributions for each pair of images as similar and as confident as possible

while simultaneously scattering the assignments for different pairs over all possible clusters.

3.1.2 Learning Objective for Segmentation

By now, we have only introduced the mutual information as an objective for clustering. With certain modifications that were also proposed in [22], we can apply it to semantic image segmentation as well. The basic idea for that is to treat image segmentation as image classification or clustering on patches that are defined by the receptive field of the CNN. That is, we consider a CNN that takes images of spatial size $h \times w$ as an input and produces a segmentation map of the same size. Then, for each pixel of the output, the label corresponds to the assigned cluster of that pixel's receptive field. So in principle, we can maximize the mutual information for a pixel in the output of an image and its corresponding pixel in the output of the transformed image.
However, the pixels that correspond to each other are not in the same locations in the segmentation output if geometric transformations are applied. For instance, let $\mathbf{x}$ be an image and $\tilde{\mathbf{x}}$ be a horizontally flipped version of it. Then, $\Phi(\mathbf{x})$ is of the same spatial size as $\mathbf{x}$ and the pixel at position (i, j), i.e. $\Phi(\mathbf{x})_{(i,j)} \in [0, 1]^C$, indicates the cluster assignment distribution for the patch centered at (i, j). In contrast to that, the pixel $\Phi(\tilde{\mathbf{x}})_{(i,j)}$ belongs to the location (i, j) in $\tilde{\mathbf{x}}$, which, in turn, corresponds to the location $(i, w - j)$ in the original image $\mathbf{x}$, where w denotes the image width. This holds as $\tilde{\mathbf{x}}$ was horizontally flipped. Therefore, maximizing the mutual information of the output pixels $\Phi(\mathbf{x})_{(i,j)}$ and $\Phi(\tilde{\mathbf{x}})_{(i,j)}$ is not sensible as they are generally not related to each other. Instead, we are interested in maximizing $I\left(\Phi(\mathbf{x})_{(i,j)}, \Phi(\tilde{\mathbf{x}})_{(i,w-j)}\right)$ because these two pixels represent the same location in the original image.
That is, we have to undo the geometric transformations before maximizing our objective to make sure that the output locations match semantically. Formally, this can be denoted with $\max_\Phi I\left(\Phi(\mathbf{x}), g^{-1}(\Phi(g(\mathbf{x}))\right)$, where $g(\cdot)$ is a geometric transformation and $g^{-1}(\cdot)$ is its inverse.
On top of that, it is proposed in [22] to add *local spatial invariance*, i.e. to not only compare pixels representing the exact same locations, but also neighboring pixels. The reasoning behind that is that if a pixel belongs to a certain cluster, the neighboring pixels are also more likely to belong to that cluster. In essence, this adds a constraint on spatial continuity in the segmentations produced by the CNN. This can be achieved by enforcing invariance with respect to small displacements, i.e. by applying the IIC objective on $\left(\Phi(\mathbf{x})_{(i,j)}, \Phi(\mathbf{x})_{(i+t_1,j+t_2)}\right)$. Here, $(t_1, t_2) \in T \subset \mathbb{Z}^2$ denotes a small translation, where T represents the set of allowed translations.
Putting all these pieces together, we obtain the IIC objective for segmentation

as

$$\max_{\Phi} \frac{1}{\mid T \mid} \sum_{(t_1,t_2)\in T} I(\mathbf{P}_{(t_1,t_2)}) \,, \tag{3}$$

where

$$\mathbf{P}_{(t_1,t_2)} = \frac{1}{n \mid G \mid\mid \Omega \mid} \sum_{k=1}^{n} \sum_{g\in G} \underbrace{\sum_{(i,j)\in\Omega} \Phi(\mathbf{x}_k)_{(i,j)} \cdot \left[g^{-1}(\Phi(g(\mathbf{x}_k))\right]^{\top}_{(i+t_1,j+t_2)}}_{\text{convolution}} \cdot$$

With Ω we denote the set of all possible pixel locations, i.e. $\Omega = [0, ..., h] \times [0, ..., w]$, and with G the set of random geometric transformations. Note that one can also add photometric augmentations like color jittering to each $g \in G$ without having to reverse them in g^{-1}.
The authors of [22] mention that taking the expectation over T could also be performed before calculating the mutual information. Nonetheless, their experiments showed that taking the expectation in the last step leads to slightly better results. They also provide an efficient implementation of (3) where the innermost sum over Ω corresponds to a 2D-convolution.

3.1.3 Architecture and Training

The authors of [22] use ResNet [18] and VGG11-like [40] CNNs as feature extractors and add a head for the generation of the output on top of that. This head consists of a single linear and a Softmax layer for clustering or a single 1×1-convolution together with a Softmax layer for segmentation. To increase the robustness, they propose to add multiple replicas of the head which are simultaneously trained such that at the end, the best performing output head can be chosen and used for testing and inference.
Furthermore, they report that *auxiliary overclustering* is highly beneficial for their approach. That is, they simultaneously train an additional overclustering head, which predicts significantly more classes than the actual number of ground-truth classes. This should help to increase the expressivity of the feature extractor. For optimization, the well-known Adam optimizer [26] can be used. The main and the auxiliary head are trained in alternate epochs. The overall framework is summarized in Figure 2.

3.1.4 Discussion

Now, we will have a look at the advantages and drawbacks of the IIC approach. A major strength of this method is that it is end-to-end trainable and does not require any kind of postprocessing. We can directly optimize our model with

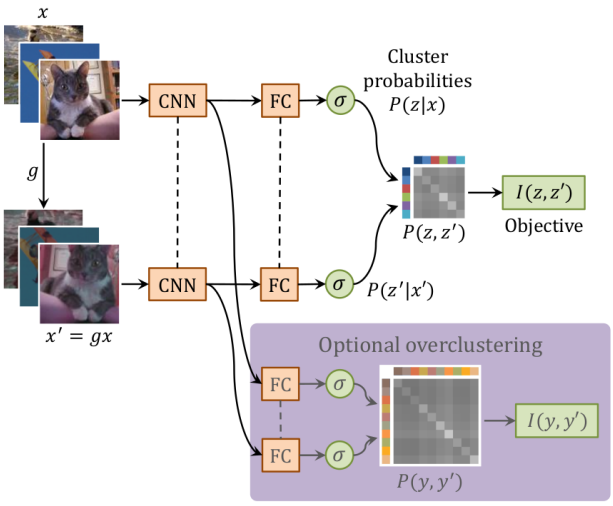


Figure 2: Framework for image clustering with IIC. Pairs of images are generated with random geometric and photometric transformations g. Then, a CNN extracts feature representations which are used by the output head to predict a distribution over clusters for each image. Finally, the mutual information criterion of equation (2) (or equation (3) for segmentation) is applied to both the main and the overclustering output. Dashed lines indicate shared weights. *Source:* [22].

respect to the learning objective, which gives us a model that outputs a probability distribution over the clusters. Unlike many other methods, there is no need to apply a separate clustering algorithm on some learned representations, which is highly convenient.
Additionally, the learning objective avoids degenerate solutions by design as we explained in depth earlier. This does not hold for all of the comparable approaches, which often require a careful choice of hyperparameters.
However, the most important point to mention here is that IIC performs very well in practice, setting a new state of the art on some benchmark datasets. The authors report the scores listed in Table 1. Admittedly, the list of models used for the comparison does not comprise other methods that were specifically designed for semantic segmentation. But, this shows us again that unsupervised semantic segmentation can be considered a hard problem with few literature on it.
Also, note that the IIC objective is not specific to computer vision, but, in principle, applicable in other domains as well. So, this novel objective can be seen as a cornerstone for a whole concept in unsupervised learning.
Nonetheless, there are also downsides to this approach. For instance, we argue

Method	Coco-stuff	Potsdam
K-Means	14.1	35.3
SIFT* [30]	20.2	28.5
Doersch* [10]	23.1	37.2
Isola* [20]	24.3	44.9
DeepCluster* [5]	19.9	29.2
IIC	**27.7**	**45.4**

Table 1: Pixelwise accuracy scores for unsupervised segmentation on the Coco-stuff [4] and Potsdam [21] dataset. Values taken from [22]. Methods that do not directly learn a segmentation function, but require clustering with K-Means on pixel level in a final step are marked with *.

that the mutual information criterion as proposed in equation (2) might not be suitable for imbalanced datasets, i.e. datasets where the cluster sizes are far from equal to each other. The reason for that is that, as we already mentioned, the mutual information can be decomposed as $I(z, \tilde{z}) = H(z)+H(\tilde{z})-H(z, \tilde{z})$. Thus, maximizing $I(z, \tilde{z})$ implies maximizing the entropies $H(z)$ and $H(\tilde{z})$, which leads optimally to a uniform distribution over all the clusters for z and $\tilde{z}$. However, for an imbalanced dataset, the uniform distribution over all the clusters does not match the true cluster distribution leading to a possibly poor clustering. To overcome this problem, it might be sensible to replace the entropy terms by the Kullback-Leibler divergence to a known distribution as it is suggested in [15]. Unfortunately, this requires prior knowledge about the distribution of the true clusters which is in general not available.

3.2 W-Net

Another well-known approach in the field of unsupervised image segmentation is the so-called W-Net [45]. Its name stems from its architecture being the chaining of two U-Nets, where the first U-Nets produces the image segmentation which the second one uses to reconstruct the original input. Hence, the W-Net represents the class of encoder-decoder methods. In the following, we will first describe the architecture of the W-Net before we treat the loss functions that are used for training.

3.2.1 Architecture

U-Net Since the W-Net consists of two U-Nets, we start by introducing the U-Net. It was proposed in [38] for the task of supervised segmentation of medical images. The architecture is formed by a contracting path and an

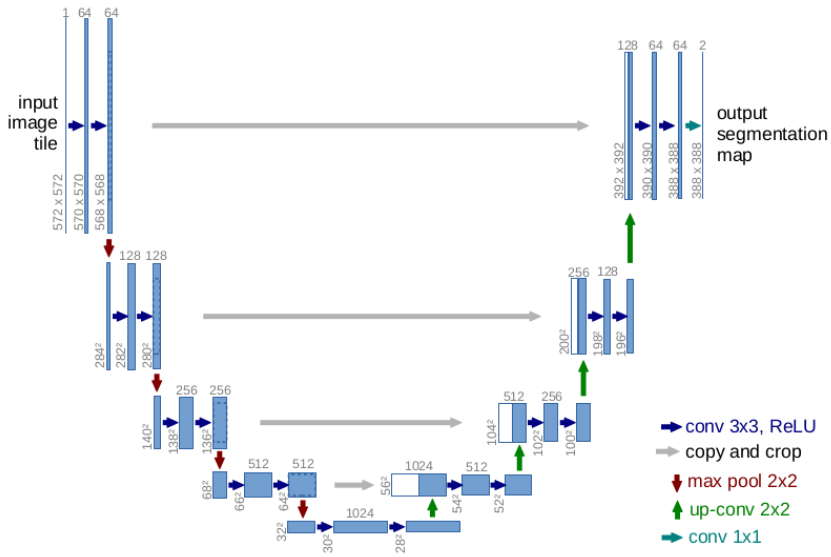


Figure 3: The architecture of the U-Net. *Source:* [38].

expanding path, which are symmetric to each other. We depict the model components in Figure 3.
The contracting path consists of four blocks with two convolutions and a max-pooling operation for downsampling. This is a pattern that occurs in the majority of recent CNN architectures and aims at extracting meaningful features. Thereby, the features become richer and more complex as we go deeper into the network. This is accompanied by the loss of spatial resolution which is necessary to make such a network computationally tractable.
On the other side, we have the expanding path aiming at reconstructing the spatial resolution of the original image for the features extracted by the contracting path. The expanding path also consists of four blocks, where all the blocks have one transposed convolution layer for upsampling as well as two regular convolution layers. Additionally, the last one uses a 1×1-convolution for producing the final segmentation map. The expanding blocks do not only process the features of their predecessors, but they are also provided with the features of their counterparts on the contracting path. This supports the exact reconstruction of the spatial resolution from deeper, but spatially less precise features. That way, the bottleneck for the spatial resolution is mitigated. At the bottom of the architecture, there is a block with two convolutions that acts as a bridge between the two paths.

W-Net The W-Net architecture with its encoder and decoder U-Net is shown in Figure 4. The U-Net architecture is basically the one described above, but

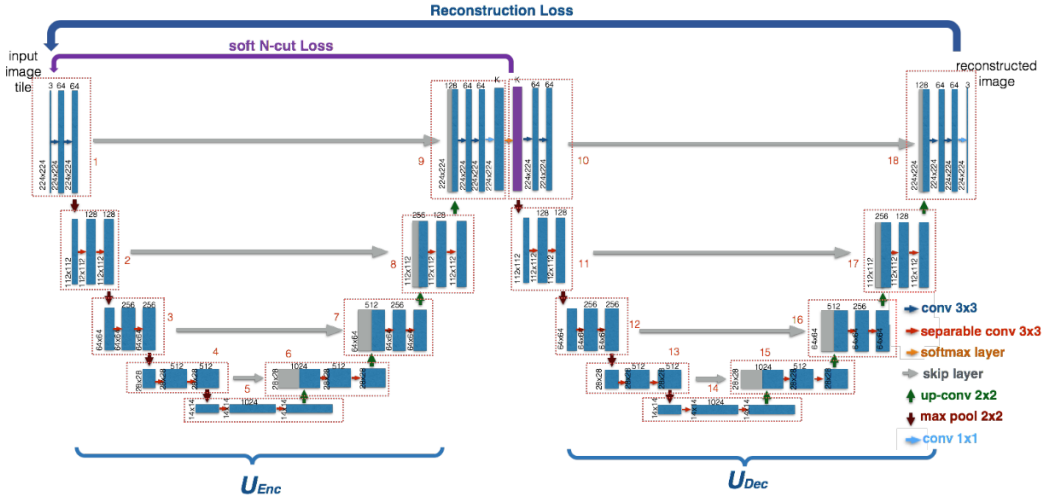


Figure 4: The architecture of the W-Net. *Source:* [45].

some minor changes were made.
First of all, the majority of the convolutions are depthwise separable as indicated in Figure 4. Depthwise separable convolutions were proposed in [8] and consist of a depthwise and a pointwise convolution. The depthwise convolution treats all the input channels separately and is followed by the pointwise 1×1-convolution which combines the extracted features from different channels without accessing any spatial relationships. This leads to a substantial reduction in computational complexity.
Moreover, all the convolutions in the W-Net are padded such that the spatial sizes of the feature maps remain the same within each block and they are only changed by the pooling operations and transposed convolutions between blocks.
Aside from that, the two U-Nets are connected by a Softmax layer. So, at the end of the encoder U-Net, we obtain a segmentation map with a cluster assignment distribution for each pixel.

3.2.2 Learning Objective

Now that we have described the W-Net and its components, the next question is how we can train it to yield useful segmentations. The loss function employed for this purpose consists of two parts, a soft N-cut loss and a reconstruction loss.

Soft N-cut Loss This loss is applied to the proposed segmentation, i.e. the output of the encoder U-Net. It is used to enforce spatial continuity within segments and a strong disassociation between segments. The concept for that was introduced in [23] and originally developed for the problem of graph partitioning. Therefore, we consider our input image as a graph $G = (V, E)$ with the nodes V being the set of pixels in the image. The edge weights are chosen as

$$w(x_{(ij)}, x_{(kl)}) = \exp\left(\frac{- \parallel x_{(ij)} - x_{(kl)} \parallel_2^2}{\sigma_I^2}\right) \cdot \begin{cases} \exp\left(\frac{-\|(i,j)-(k,l)\|_2^2}{\sigma_X^2}\right) & \text{if } \parallel (i,j) - (k,l) \parallel_2 < r \\ 0 & \text{otherwise.} \end{cases}$$

That is, two pixels $x_{(ij)}$ and $x_{(kl)}$ are not connected by an edge if their spatial distance is larger than some threshold r. In the opposite case, their edge weight is a positive real number such that both a small dissimilarity in pixel values $\parallel x_{(ij)} - x_{(kl)} \parallel_2^2$ as well as a small spatial distance $\parallel (i,j) - (k,l) \parallel_2^2$ lead to a larger value, i.e. a stronger association. Furthermore, σ_I^2 and σ_X^2 act as scaling parameters.
Given this graph G with its edge weights, the normalized cut criterion of [23] measures how well a segmentation S partitions G and the corresponding image. It is defined as

$$\mathcal{L}_{N\text{-}cut}(G, S) = \sum_{c=1}^{C} \frac{\sum_{u \in A_c, v \in V \setminus A_c} w(u, v)}{\sum_{u \in A_c, t \in V} w(u, t)}. \tag{4}$$

Here, the segmentation $S = \{A_1, ..., A_C\}$ is a partition of V, i.e. $V = \dot{\bigcup}_{c=1}^{C} A_c$. So, S consists of hard pixel labels $c \in \{1, ..., C\}$ and A_c is the set of pixels assigned to c.
As small weights on edges between pixels with different labels (numerator) and, simultaneously, large weights on edges between pixels with the same label (denominator) make the fraction small, a small value for the N-cut indicates a good segmentation. To apply equation (4) to the segmentations produced by the W-Net, we would have to perform the argmax-operation on the output of the encoder U-Net in order to turn the Softmax scores into a hard labeling. Unfortunately, the argmax-operation is not differentiable which prohibits us from training with gradient descent.
Therefore, the authors of [45] propose a soft, differentiable version of equation (4). They define it as

$$\mathcal{L}_{soft\text{-}N\text{-}cut}(G, S) = C - \sum_{c=1}^{C} \frac{\sum_{u \in V, v \in V} w(u, v) P(u \in A_c) P(v \in A_c)}{\sum_{u \in V, t \in V} w(u, t) P(u \in A_c)}. \tag{5}$$

In this formulation, the segmentation S provides a discrete distribution over the clusters $1, ..., C$ for every pixel $v \in V$. So, we can directly plug the Softmax probabilities of the encoder U-Net into equation (5). Thus, we do not have to extract hard labels making this loss differentiable and suitable for gradient-based optimization. Besides that, the functioning of the soft-N-cut loss is very similar to its non-differentiable version. It is minimized when pairs of pixels with a high similarity, i.e. a large weight on the edge between them, have high probabilities to belong to one shared cluster, whereas large edge weights for pixels that are likely to belong to different clusters increase the loss.

Reconstruction Loss To make the segmentations of the encoder U-Net semantically meaningful, we have to set them in relation to the input image. This is done in a typical manner for encoder-decoder architectures by reconstructing the input image from the segmentation and applying a reconstruction loss on the reconstructed image. For the W-Net, the reconstruction loss is chosen to be the L2 loss, i.e.

$$\mathcal{L}_{reconstr} = \| \mathbf{x} - U_{Dec}(U_{Enc}(\mathbf{x})) \|_2^2 . \tag{6}$$

Training In summary, the soft-N-cut loss in equation (5) is applied to the segmentation to enforce a consistent and distinctive segmentation, while simultaneously the reconstruction loss in equation (6) acts on the reconstruction. This ensures a direct relation between the proposed segmentation and the original image.
Both loss functions are minimized at the same time during training. In particular, the soft-N-cut loss (5) of the Softmax output of U_{Enc} is used to update the encoder. Additionally, the reconstruction loss (6) applied to the output of U_{Dec} produces a gradient for both the encoder and the decoder.

3.2.3 Postprocessing

In [45], it is reported that the segmentations of the W-Net require some post-processing to unfold their full potential. It is proposed to use a combination of conditional random fields and hierarchical merging.

Fully-Connected Conditional Random Fields for Edge Recovery A common issue in computer vision with deep learning is that multiple pooling operations, as we see them in the U- and W-Net, lead to a loss of localization accuracy. The skip connections of the U-Net aim at mitigating this problem, but in addition, a fully-connected conditional random field (CRF) [6, 45] can be used to further improve the results. The main idea of CRFs is to optimize

an energy function which imposes smoothness constraints on the labeling of similar pixels. Thus, the application of a CRF to the output of the encoder U-Net leads to a smoother segmentation with less spurious regions. However, we do not want to go into more detail here since we did not use CRFs for our task (see Section 5.3.1).

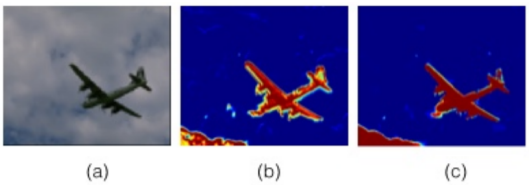


Figure 5: The effect of CRF smoothing. (a) shows the original input image, (b) the segmentation proposed by the W-Net, and (c) the segmentation after the application of a fully-connected CRF. *Source:* [45].

An example of the effect of CRF smoothing can be seen in Figure 5. Clearly, the output of the CRF has sharper boundaries and constitutes a more consistent segmentation.

Hierarchical Merging Unfortunately, the results after postprocessing with CRFs tend to be oversegmented and some segments should be merged to form larger ones. The authors of [45] accomplish this by producing a hierarchical segmentation based on pixel features on segment boundaries. Once again, we omit the details here as we did not find this kind of postprocessing to be reasonable for our work (see Section 5.3.1). In Figure 6, we show the results of both postprocessing steps in combination, which is also the final output as proposed in [45].

3.2.4 Discussion

To conclude this section, we will discuss some of the properties of the W-Net. First of all, it is a relatively simple and intuitive approach for image segmentation. The fact that the encoder U-Net directly produces a segmentation with probability scores makes the training quite handy, even if this is not the final result. The novel soft-N-cut loss is a useful tool to produce well-partitioned segmentations, which might also be suitable in other scenarios than training a W-Net. As it is differentiable, it can be potentially integrated into any end-to-end training framework.
Moreover, the experiments of [45] show that the segmentation results obtained from the W-Net are both qualitatively and quantitatively promising. For instance, it is reported that the W-Net outperforms all other approaches on

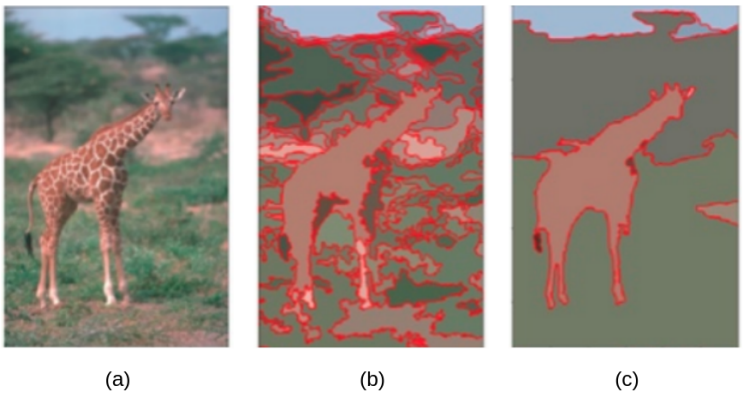


Figure 6: CRF smoothing and hierarchical merging in combination. (a) shows the original input image, (b) the output of the CRF, and (c) the final segmentation after hierarchical merging. *Source:* [45].

the Berkeley Segmentation Dataset (BSDS) [31]. Nevertheless, we have to remark that the authors do not compare their method with other deep learning approaches. Instead, their benchmark approaches are rather traditional computer vision methods like [1].
It is also important to note that the W-Net is proposed as a model for non-semantic image segmentation. However, we argue that the design with a Softmax output indeed assigns similar objects to the same clusters. Therefore, segments with the same label can be expected to share a semantic concept. This is especially the case for a dataset like ours where the objects of interest are not composed of multiple complex features.
A drawback of the W-Net is that additional postprocessing of the segmentation maps is necessary. Thus, the proposed model is not entirely end-to-end trainable and including the postprocessing mechanisms directly into the training would be clearly desirable. In addition to that, the architecture is quite large. Although this was alleviated a little bit with depthwise-separable convolutions, the model with its two U-Nets remains compute-intense. However, only for training, we need both U-Nets. After training, the decoder U-Net can be discarded as it is not involved in producing the segmentation maps.

3.3 Unsupervised Visual Representation Learning by Context Prediction

An approach for unsupervised learning on images in general was proposed by Doersch et al. in [10]. The idea of their work is to learn semantic concepts by predicting the spatial relation of image patches. Although this method was in

principle not designed for semantic image segmentation, the architectures can be adjusted such that input-sized feature maps are produced which can then be turned into a segmentation map by pixel clustering. This was done in [22] and the results show that such an approach can work decently.

3.3.1 Learning Framework

Context Prediction In order to use context as a valuable source of information for unlabeled image data, the original images are divided into equally sized patches as depicted in Figure 7. Then, an arbitrary patch is fed into a CNN together with a randomly selected neighboring patch. The CNN has to predict the location of the second patch relative to the first one, i.e. top-left, top, top-right and so on.

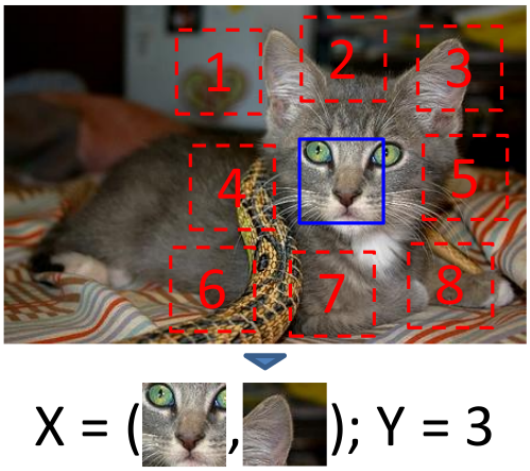


Figure 7: Sample image divided into patches. The model input X is the pair of the center patch and a randomly selected second patch. The target Y is a label representing the spatial relation of the input patches. *Source:* [10].

In doing so, the network is expected to learn representations of concepts contained in the data. For instance, if we consider the example in Figure 7, the model might eventually learn the concept of cat eyes and cat ears as these occur in a certain spatial context most of the time (ears are located above and slightly horizontally shifted with respect to eyes). Thus, the spatial context can be leveraged as an informative supervisory signal in unsupervised learning.

Architecture and Training The architecture of the network predicting the targets is a so-called *late-fusion architecture* [10] as it is depicted in Figure 8. The reasoning behind that is that representations should be learned for individual patches and not for pairs of patches. So, the feature extractor CNN (conv1

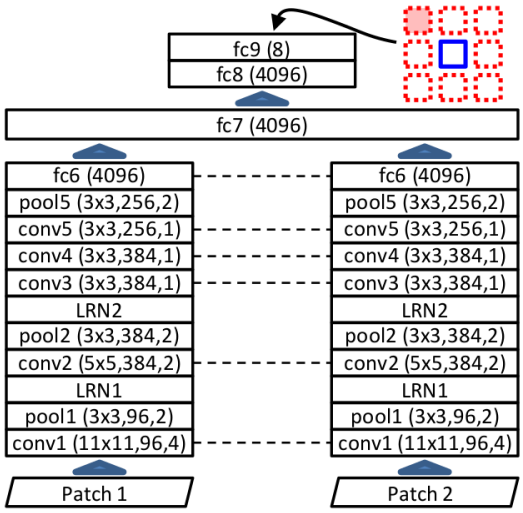


Figure 8: Late-fusion architecture consisting of a CNN as a feature extractor and an output head with three fully-connected layers. 'LRN' denotes local response normalization layers and dashed lines indicate shared weights. *Source:* [10].

to fc6) acts on the patches independently and produces the representations we are ultimately interested in. Only for training, we employ the classification head, which allows us to infuse the context information into the network and the extracted representations. We have also seen this architecture style in Section 3.1.
As the task of predicting the correct spatial relation between patches is an ordinary classification problem, we can train the described network with the cross-entropy loss. However, the authors of [10] do not explicitly state their choice of a loss function.

Trivial Solutions When training the described model, it is important to make sure that there are no cues in the data that can serve as a trivial shortcut. This could lead to the fact that the network does not have to learn the desired concepts, but only exploits those cues and finds trivial solutions.
In our case, such cues can be patterns and textures like edges at the boundaries of patches. For example, an edge that starts on the right-hand side of a patch and is perfectly continued on the left boundary of a second patch is in most of the cases enough information to say that these patches are located next to each other. But, we are not interested in low-level features such as edges at boundaries but rather semantic concepts. Therefore, the authors of [10] introduce a gap between the patches and, additionally, randomly displace them

by a small number of pixels (see Figure 7).
Another cue that allows for trivial solutions is far less obvious. In the paper, it is reported that CNNs are apparently able to learn the position of the patch relative to the camera lens due to *chromatic aberration*. This is the phenomenon that a color channel shrinks from the image boundaries towards the center. As a consequence, the relative position of two patches can be easily inferred from their relative position to the lens. To overcome this, the authors propose to either project the color values to the orthogonal space of the green-magenta axis ($[-1, 2, 1]$ in the RGB space) or to randomly drop two of the three color channels and replace them with noise in a preprocessing step.

3.3.2 Adjustments for Segmentation

We have only treated this approach for the purpose of unsupervised feature learning by now. However, as stated in [22], it can be extended to make it applicable for unsupervised semantic segmentation. Therefore, a feature vector has to be extracted for every pixel. This feature vector can be considered as the representation of the patch defined by the receptive field of the particular pixel. When feeding pairs of feature vectors into the output head, it is important to make sure that their receptive fields do not overlap.
Another point to mention here is that the CNNs used in [22] lead to less precise feature maps and segmentations. The reason for that is that pooling operations in CNNs substantially reduce the spatial resolution of feature maps throughout the layers. Thus, to obtain a representation map of the same spatial size as the input image, we have to perform an upsampling of the representations at a lower resolution. Unfortunately, this implies that the resulting feature and also the segmentation maps are less precise. Considering a feature extractor without pooling operations, i.e. without decreasing the resolution, is computationally intractable.
Once we trained a CNN to produce meaningful feature maps with the same spatial size as the input images, we have to convert them into a segmentation map. This can be achieved with K-Means clustering, i.e. each data point is a feature vector of one pixel and these data points are assigned to different groups by the clustering algorithm. To decrease the computational cost, it is also possible to cluster the representation vectors before the upsampling step and upsample the obtained segmentation maps afterward.

3.3.3 Discussion

The approach of employing context as a supervisory signal for unsupervised representation learning is reasonable and appealing. Furthermore, it has also proven itself successful on different occasions. This is also the case when it

comes to clustering such representations. In [10], the authors report that the representations obtained by this method fulfill the desirable property that representations of semantically similar images are also close in the embedding space. Some illustrative examples are shown in Figure 9. This suggests that the representations are suitable for dividing the data into semantically meaningful clusters.

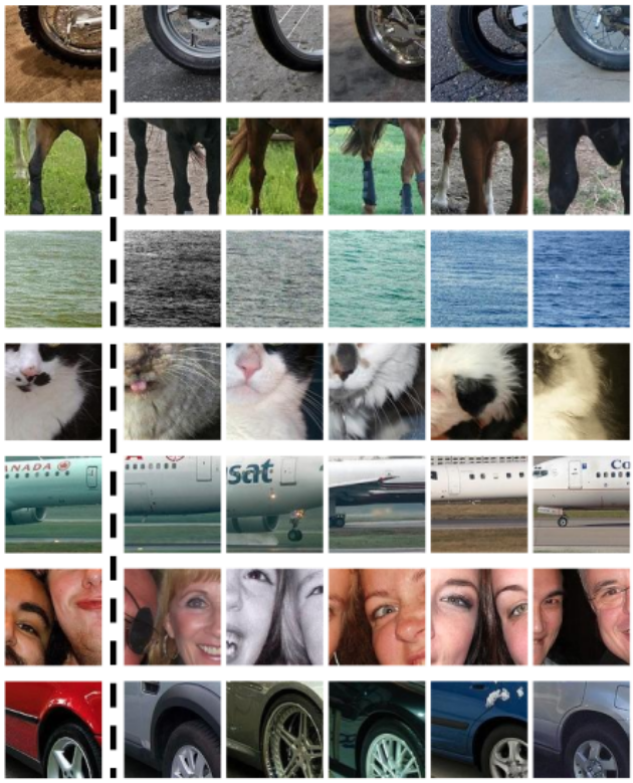


Figure 9: Nearest neighbors based on representations learned via context prediction. On the left side of the dashed line is an input image and on the right side in the same row are images whose representations are nearest neighbors of the inputs representation. *Source:* [10].

On top of that, [22] report that leveraging this learning procedure for semantic segmentation results in a decent performance on the Coco-stuff [4] and Potsdam [21] dataset (see Table 1).
We also reason that feature learning by context prediction is sensible for our specific task since coarse woody debris often has a large spatial extent compared to other frequent objects in our dataset, e.g. standing trees. Hence, it may play an important role in context prediction and, therefore, the learned features may capture coarse woody debris well.

But, there are also downsides to this method. Essentially, it was designed for representation learning and not for clustering or even unsupervised semantic segmentation. Hence, a segmentation model can not be trained in an end-to-end fashion. Instead, we have to fit a clustering algorithm like K-Means in a second step.
Moreover, since the clustering happens on pixel level, there are no typical segmentation constraints integrated. For instance, the soft-N-cut loss in the W-Net ensures that the segmentation is distinctive but consistent at the same time. This approach lacks such mechanisms, which is why the produced segmentation maps can be somewhat erratic and fragmented (see Section 5.4.2).

3.4 Others

Although the number of existing approaches for unsupervised image segmentation is relatively small - especially in comparison to other well-studied problems in computer vision - there are some approaches we did not pursue in this work. For the sake of completeness, we will briefly describe some of the more prominent examples and explain why we decided against implementing them.

3.4.1 Learning Visual Groups From Co-Occurrences in Space and Time

This technique proposed in [20] is conceptually quite similar to [10], which we already discussed in Section 3.3. Instead of context prediction, this method is based on proximity prediction.

Idea In order to employ proximity as a supervisory signal for self-supervised learning, pairs of neighboring image patches are fed through a feature extractor CNN. Then, a binary classification head predicts whether the two patches are adjacent or not. To balance the training, negative samples are generated by taking pairs of patches from random locations. By backpropagating a classification loss, the network can be updated to extract better features and make better predictions about proximity.
In [20], a graph is constructed from an image such that each patch from a grid corresponds to one node in the graph. Then, the edge weights are chosen as the estimated probability that the two patches are adjacent. Ultimately, applying spectral clustering on the graph yields components which correspond to object proposals.
In contrast to that, in [22], the features learned by proximity prediction are clustered with K-Means to produce a segmentation mask.

Discussion Similar to [10], this method represents an intuitive and useful concept of self-supervised learning, which is also extendable to downstream tasks like clustering. Moreover, it is transferable to other domains where a notion of proximity is available, e.g. movies with temporal proximity.
Nevertheless, it suffers from the same drawback as [10] when employed for segmentation. Concretely, no clustering function is learned directly and, therefore, the method is not explicitly designed and optimized for that purpose.
Aside from that, we argue that this method is also less suitable for our problem than [10]. The reason for that is that the vast majority of objects in our dataset is rather small compared to both the image size and the pixel size (i.e. images consist of many objects, but objects consist of relatively few pixels). As a consequence, the generation of negative samples by selecting patches from random locations may be problematic as the chance that the same semantic concept is represented in two randomly sampled patches is quite high. For example, there is a relatively high probability that two selected patches contain both a tree or both ground.

3.4.2 Unsupervised Image Segmentation by Backpropagation

The framework proposed in [25] is a representative for iterative methods. More precisely, the two subproblems of label prediction and network parameter learning are addressed in an alternating manner.

Idea In this approach, an image is fed into a CNN which directly outputs a pixelwise segmentation. After that, the output is used to generate pseudo-labels which are used to update the network. The process is repeated until a stopping criterion is met.
In doing so, three constraints are imposed on the segmentation: pixels with similar features should be assigned to the same cluster, segments should be spatially continuous and the number of unique labels should be large.
While the first constraint is naturally satisfied by CNNs due to their structure, the other two require some additional measures. Spatial continuity is enforced by generating the pseudo-labels for backpropagation on the basis of superpixels. That is, the pseudo-label for a particular pixel is the most frequently predicted label for the superpixel containing the pixel. Hence, all pixels in a superpixel receive the same supervisory signal eventually leading to spatially continuous predictions.
Furthermore, it is crucial to integrate a mechanism that keeps the number of different predicted labels high. Otherwise, the first two constraints would lead to the degenerate solution where each pixel is assigned to the same cluster. To accomplish that, it is proposed to perform intra-axis normalization in the

form of batch normalization on the final output (before applying the argmax-operation to obtain labels). By that, each output channel has zero mean and unit variance, which gives each label the same chance of being the maximum value for a certain location.

Discussion An advantage of this approach is that it does not only define sensible objectives for a good segmentation, but it also provides simple and efficient means to achieve them. Therefore, the method itself is quite graspable and can possibly be extended or adjusted in certain situations.
However, this method is not applicable for semantic segmentation because the produced segments do not represent semantic concepts but rather image regions. In fact, according to the official implementation[1], the model is trained per image until the number of found segments surpasses a predefined value. Hence, we can not apply this method to solve our problem.

3.4.3 Joint Unsupervised Learning (JULE)

The framework of Joint Unsupervised Learning, abbreviated as JULE, was originally proposed by [46] as a concept for learning deep representations as well as image clusters. Based on that, the work of [32] extended this iterative method and transferred it to the task of unsupervised segmentation of medical images.

Idea In [46], the authors use clustering labels as a supervisory signal, which in turn can be utilized for obtaining better representations of images. These two steps of clustering in the forward pass and improving representations in the backward pass are then integrated into a recurrent learning process, which yields better and better results over time.
In particular, in every training round, they feed images as input into a CNN to get some initial representations. After that, one step of agglomerative clustering is performed. That is, the two clusters of representations from the previous iteration that have the highest affinity are merged into one cluster such that new clustering labels are obtained. These are then used to compute a loss, which allows us to update the weights through backpropagation. This process is repeated until the desired number of clusters is reached. As a result, we obtain a CNN which is able to extract rich representations of the images as well as a partitioning of the set of the images into a fixed number of clusters. The whole learning framework is illustrated in Figure 10.
In [32], the authors pick up this machinery and apply it to segmentation. They do this in a way that we have already seen in other methods, i.e. by extracting

[1] `https://github.com/kanezaki/pytorch-unsupervised-segmentation`

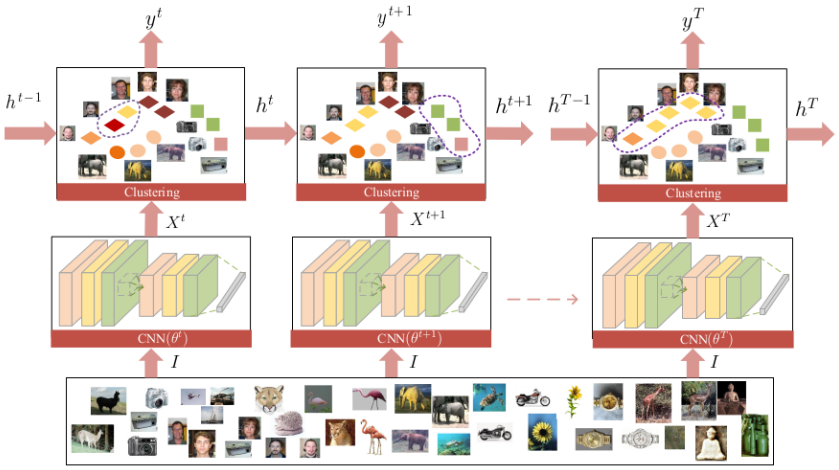


Figure 10: The recurrent learning process of JULE. *Source:* [46]

representations for pixels by associating each pixel with a corresponding patch in the input image. Furthermore, they use K-Means clustering to produce their final pixel clustering yielding a segmentation map.

Discussion It is reported in [32] that JULE works well and robustly in different situations and that the learned representations are particularly suited for clustering. However, we have decided against implementing this approach due to the following reasons. First of all, this approach being originally designed for image clustering and representation learning has the same drawbacks as other representation learning methods adapted for image segmentation (see Section 3.3.3). So, we can not expect a major performance improvement compared to other representation learning techniques applied for segmentation. On top of that, the idea may seem quite simple at first glance, but the training framework as a recurrent process is actually rather complex. In this regard, other methods are much easier to handle.

3.4.4 Emergence of Object Segmentation in Perturbed Generative Models

This approach, which was proposed in [2], belongs to the class of methods based on generative adversarial networks (GANs). Here, the generator creates binary segmentation masks separating a foreground object from the background.

Idea The framework consists of three parts: an encoder E, a generator G and a discriminator D. The job of the encoder is to map an image to a representation $\mathbf{z} \in \mathbb{R}^k$, which is then fed into the generator. The generator uses

$G(\mathbf{z})$ to create a threefold output, i.e. $G(\mathbf{z}) = [G_B(\mathbf{z}), G_F(\mathbf{z}), G_M(\mathbf{z})]$, where $G_B(\mathbf{z})$ and $G_F(\mathbf{z})$ are the background and the foreground of the original image respectively. In addition to that, $G_M(\mathbf{z}) \in [0,1]^{H \times W}$ is a mask with continuous values indicating which pixels constitute the foreground. During training, a binarization loss imposed on $G_M(\mathbf{z})$ enforces that the mask values are close to zero or one, but desirably not in the vague area around 0.5. This aims at making the mask as distinctive as possible.
Then, the three outputs of G are composed to one image again and the discriminator classifies whether the composite image is real or fake. The crucial trick that is applied here is to add small random shifts when composing the image from the proposed fore- and background. So, if the foreground object is not properly separated from the background, the perturbations lead to an unrealistic composite image which can be easily detected by the discriminator. For instance, in such a scenario, the foreground object may be cut into two separate pieces or parts of it may occur twice. We illustrate an example for that in Figure 11.

Figure 11: The trivial solution with the invalid segmentation mask in the first row leads to a realistic composite image. However, a random shift in the composition step reveals the improper segmentation. In the second row, the mask is correct and yields a realistic composite image, even with a random displacement. *Source:* [2].

The training of this framework is conducted in two phases: First, the generator and the discriminator are trained against each other. After that, the generator is fixed and the encoder is trained to act as an autoencoder together with G.

Discussion By training the generator to make the decisions of the discriminator invariant under small shifts, an implicit definition of objects in images is used. That is, an object is an object if one can displace it by a small margin and the resulting image remains realistic. In this regard, the idea of [2] is quite unique and interesting.
A drawback of this approach is that it is designed exclusively for the task of segmenting foreground objects. In a dataset of images with no or multiple

foreground objects, this may be problematic. Moreover, this method can only be used for binary segmentation and it is not really aware of the semantic meaning of its segments. Instead, only the concept of a foreground object is learned. Therefore, we reason that this approach is less appropriate for our task.

3.4.5 Deep Multi-Class Segmentation Without Ground-Truth Labels

The technique for the segmentation of medical images that is proposed in [24] is another example of segmentation methods based on adversarial learning. But, it also exhibits the characteristics of encoder-decoder methods as we will explain in the following.

Idea In this framework, the generative model is a U-Net-like segmentor that produces not only a segmentation map for the input but also some residual information that is used later. The segmentations are scrutinized by a discriminator which forces the segmentor to generate more and more realistic segmentation maps. The discriminator is trained with the proposed segmentations as well as with real segmentations from another dataset with the same anatomy (cardiac image segmentations in this case).
A severe problem that occurs if only these model components are used is that the generated segmentations do not have to be spatially aligned with the input images. In fact, the generator is only trained to produce realistic segmentations which do not have to be in any relation to the inputs. Therefore, a reconstructor additionally restores the original images from the segmentations and their corresponding residuals. A reconstruction loss is finally applied to the reconstructed and the original image. Hence, this framework is also an encoder-decoder architecture. Aside from that, two other cost functions are utilized during training to ensure that the segmentations meet the desirable properties of spatial continuity and proper delineation.

Discussion This method is another example showing that adversarial learning can be successfully applied to the domain of image segmentation. However, it is not a fully unsupervised method. Although this is suggested by the title, the authors state that they use ground-truth segmentations from another dataset for the adversarial training. Thus, this approach rather belongs to the field of domain adaption than to unsupervised image segmentation.
Unfortunately, suitable ground-truth annotations from a different but sufficiently similar dataset are often not available. This is also the case for our problem which is why we can not apply this method in our experiments.

4 Semantic Image Segmentation as a Two-Step Procedure

4.1 Motivation and Idea

When experimenting with methods of the previous section, we found out that the small and low-key segments of coarse woody debris were mostly not detected properly, i.e. the logs lying on the ground were simply treated as some kind of noise in the ground segments (see Section 5.4.2). When we relaxed the constraints on spatial continuity of the segmentation in order to be able to capture such fine segments like logs, this did not have the expected effect and, additionally, led to another downside: The segmentation maps became more fragmented and resembled sometimes more a pixelwise clustering than a semantically meaningful partitioning of the image. We reason that this is caused by the fact that allowing smaller segments eventually leads to less inclusion of high-level features and, therefore, to less meaningful segments.

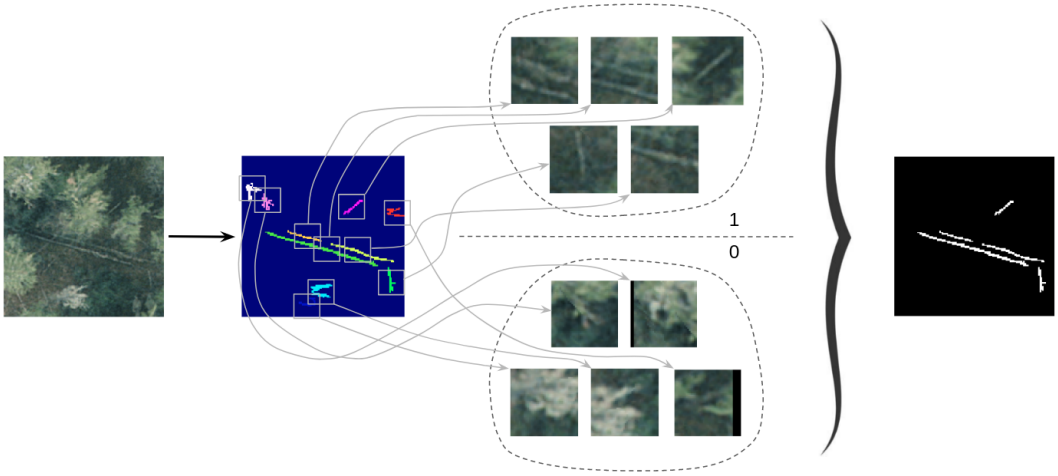


Figure 12: Overview of our proposed two-step procedure for semantic image segmentation. First, a segmentation algorithm partitions an image into different regions or segments and, then, these are assigned to semantic groups.

In order to overcome this, we broke up the process of semantic segmentation into two phases instead of training one CNN that produces pixelwise assignments. In the first step, we use an algorithm to divide the image into different segments. This is similar to the extraction of image objects in the GEOBIA framework (see Section 2.1). But, in the second step, these segments are not treated as image objects with some attributes. Instead, we extract image patches containing the segments and cluster them based on their content.

This has the advantage that we do not lose as much information as when only a predefined set of attributes like channel means and standard deviations is utilized. We argue that such representations may suffice to detect relatively simple features like coarse woody debris, as demonstrated in [28], but rather complex features require more information that can only be provided if we treat segments as what they actually are: parts of an image.
Altogether, one can say that we divide the problem of semantic segmentation into two subproblems being non-semantic image segmentation and assigning semantic labels to the segments. We depict an overview of this concept in Figure 12. In the following, we will describe how the two steps for our approach can be realized.

4.2 Segmentation Algorithms

4.2.1 Felzenszwalb's Efficient Graph-based Image Segmentation

The segmentation algorithm proposed by Felzenszwalb and Huttenlocher [13] is based on graph partitioning. More precisely, an image is considered as a graph $G = (V, E)$, where each pixel is a vertex $v \in V$ and the edges $(v_i, v_j) \in E$ are pairs of neighboring pixels. Furthermore, every edge has a weight $w((v_i, v_j)) \in \mathbb{R}_+$ which indicates the dissimilarity of the two pixels based on features like difference in brightness or color.
In this setting, a useful segmentation of an image corresponds to a partition of the graph vertices into disjoint components such that the dissimilarities are large between components and low within components. Thereby, the dissimilarity of two components is measured by the minimal dissimilarity between pairs of connected vertices in the two components.
The partition is obtained by a bottom-up procedure, where, in the beginning, each vertex is its own component. Subsequently, a loop over the edges in the order of ascending weights is performed. In every step, two components are merged if the vertices connected by the current edge are in different components and a dissimilarity-based merging criterion is met.
A benefit of this method is that it only requires one loop over the edges making it computationally efficient. Moreover, by modifying the parameters of the edge weight function, the granularity of the segmentation can be set to the desired level. This is especially useful for small objects like coarse woody debris.

4.2.2 Edge-based Segmentation with DBSCAN

As the vast majority of segments corresponding to coarse woody debris have very similar and relatively simple features, we were able to design a segmenta-

tion algorithm that is specialized in coarse woody debris. The initial observation for this was that coarse woody debris very often occurs on straight edges in the image. Thus, by extracting groups of pixels lying on long, straight edges, one can obtain segments that serve as candidates for coarse woody debris.

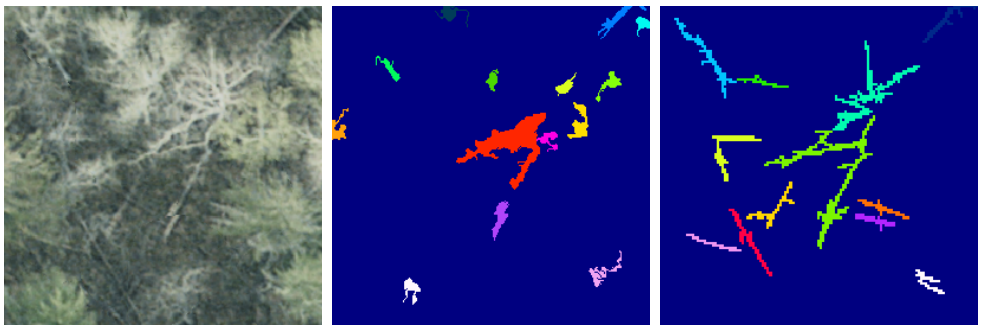
Detection of Straight Edges To detect such edges, we used a modified version of the Prewitt operator [36]. Instead of having only two 3×3 filter matrices detecting horizontal and vertical edges, we use several 25×25 filters to detect edges at different angles. The increased filter size allows to put a focus on the long and straight edges of logs.
The filter corresponding to horizontal lines is shown in the matrix below.

$$\begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \cdots & & & \vdots \\ -1 & \cdots & -1 & \cdots & -1 \\ 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 \\ 2 & \cdots & 2 & \cdots & 2 \\ 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 \\ -1 & \cdots & -1 & \cdots & -1 \\ \vdots & \cdots & & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}$$

In this manner, we also create multiple rotated versions of this filter in order to ensure that the detected features are invariant under different angles. To aggregate the convolution output channels for the different angles, we use the max-operation since it leads to a more distinctive feature map than, for example, the L2 norm over the channels.
Another design choice is to not only have two rows of non-zeros like the Prewitt filter but three, i.e. two rows with the value -1 at the outside and one row with value 2 at the inside. This leads to the fact that our filter actually detects straight "double edges", which is preferable in our case as logs usually have edges on both sides. We also found it beneficial to add a second row of zeros between the rows of non-zeros.

Separation of Segments Once the straight edge activations are computed, we transform them into a binary edge map by thresholding. That is, the value one is assigned to each pixel with an activation larger than a threshold τ, and zero otherwise.

So, we obtain a set of pixels that are candidates for representing coarse woody debris. Before we can make final decisions, we partition this set of pixels into segments which are then the basis for the assignment of classes. This is done by applying the DBSCAN [12] clustering algorithm to the pixel locations, i.e. their spatial indices. By that, we obtain the segments as spatially coherent clusters.
Using DBSCAN for that has two main advantages. Firstly, it determines the number of sensible clusters or segments itself, which is necessary as the number of segments varies between different images. Aside from that, it is also able to identify noise, i.e. activated pixels without any other activated pixels in their neighborhood are ignored. This is desirable as an image segment should not consist of only one or a few pixels. However, we even tighten this by explicitly removing clusters smaller than a specified minimum size.
And secondly, DBSCAN imposes no assumptions on the shape of clusters, in contrast to other algorithms like K-Means. This is appropriate for the coarse woody debris segments as these are in general longish and not always convex.
Before the clustering step, it is also possible to reduce the computational complexity by downsampling the binary edge map, i.e. reducing the spatial resolution in order to reduce the number of samples to be clustered into segments. After that, the segmentation map can be easily upsampled again.



(a) Image (b) Felzenszwalb (c) Our method

Figure 13: Felzenszwalb's method in comparison with our edge-based segmentation. Apparently, Felzenszwalb's algorithm tends to look for entire image regions, whereas our method was specifically built for the detection of segments as edges. Hence, it is able to better capture the segments corresponding to coarse woody debris.

Summary We summarize the workflow of our algorithm for the extraction of candidate segments in Figure 14. As we already mentioned, this segmentation algorithm is specifically designed for extracting segments that are likely to

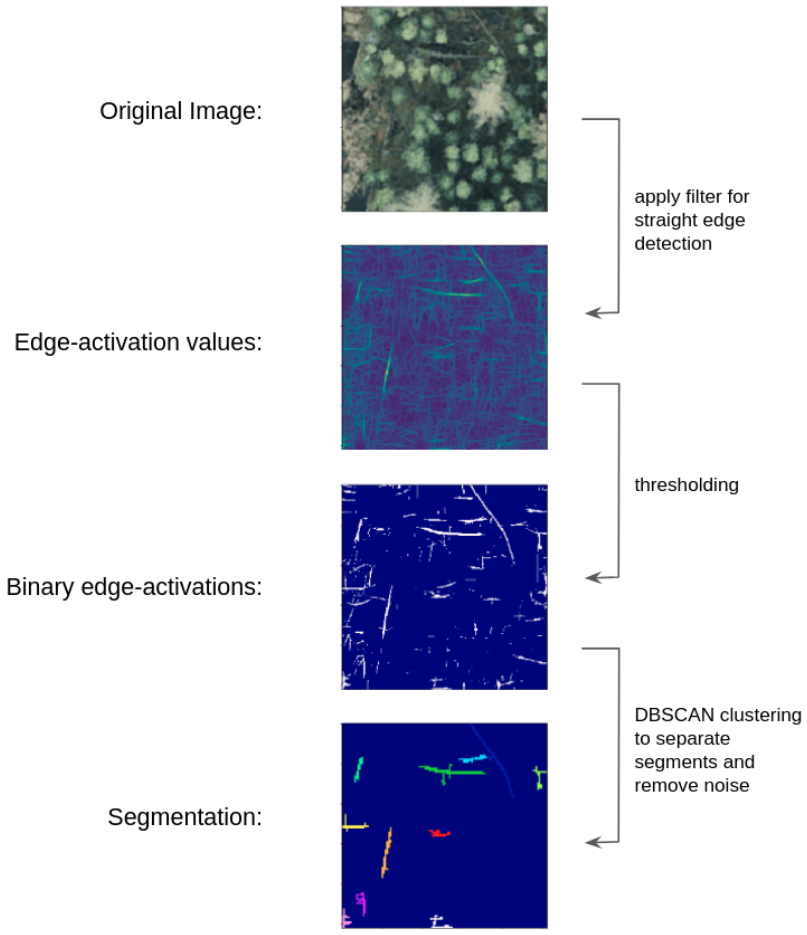


Figure 14: Workflow of our edge-based segmentation algorithm. Not only segments of coarse woody debris are extracted but also other features, like the border of the water body in the top-left corner.

contain coarse woody debris. Because of that, it is better in this regard (see Figure 13), but in general, classical methods like the Felzenszwalb's algorithm produce more universal segmentations.

4.3 Semantic Label Assignment

To provide labels for the extracted segments, i.e. to turn the initial segmentation into a semantic segmentation, a clustering algorithm is applied. In particular, we use patches containing the segments as input and perform image clustering - sometimes called unsupervised image classification - on these. The gain of using patches instead of other simpler and more condensed rep-

resentations is that patches contain the most detailed information about the features in them.
To this end, one can employ the invariant information clustering [22], which we presented thoroughly in Section 3.1. Once again, the advantages of this method are that the model is end-to-end trainable and theoretically invariant under the specified set of transformations. Hence, we can expect the algorithm to cluster the patches with respect to the features that are relevant for us and not some meaningless features like spatial orientation.
Next, we will look at some other approaches that can be used to assign meaningful labels to segments.

4.3.1 SimCLR

SimCLR, being the abbreviation for *Simple Framework for Contrastive Learning of Visual Representations*, was proposed in [7]. As the name already says, it is a framework for learning rich image representations in an unsupervised way. Similar to IIC, this contrastive learning approach aims to maximize the agreement for two randomly augmented versions of an image in order to capture the features of interest and filter out the irrelevant details.

Learning Objective The framework for this method is shown in Figure 15. First, two image versions, $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$, are created by applying some random transformations $t, t' \sim \mathcal{T}$ to an input image $\mathbf{x}$. Then, a CNN $f(\cdot)$ extracts features from $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$ leading to representations $\mathbf{h}_i$ and $\mathbf{h}_j$. These are the representations that we are ultimately interested in and that we use for downstream tasks. During training, $\mathbf{h}_i$ and $\mathbf{h}_j$ are mapped to their corresponding output representations $\mathbf{z}_i$ and $\mathbf{z}_j$ by a projection head $g(\cdot)$. Thereafter, $\mathbf{z}_i$ and $\mathbf{z}_j$ are used to generate a loss which allows to train the networks $f(\cdot)$ and $g(\cdot)$ with backpropagation.
More precisely, the similarity of the outputs is defined as $\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^\top \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|}$, which is the well-known cosine similarity. Based on that, for a positive pair of samples (i, j), i.e. $\mathbf{z}_i$ and $\mathbf{z}_j$ represent the same original image, we define

$$\ell_{i,j} = -\log \left(\frac{\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \, \mathrm{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau} \right). \tag{7}$$

Here, τ denotes a positive temperature parameter and $\mathbb{1}_{[\cdot]}$ is the indicator function that takes the value one if the term in square brackets evaluates to true and zero otherwise. The meaning of equation (7) is that, for a positive pair (i, j), all the other pairs (i, k) with $k \neq i$ in the batch serve as negative examples. Thus, by minimizing $\ell_{i,j}$, the similarity of $\mathbf{z}_i$ and $\mathbf{z}_j$ is maximized,

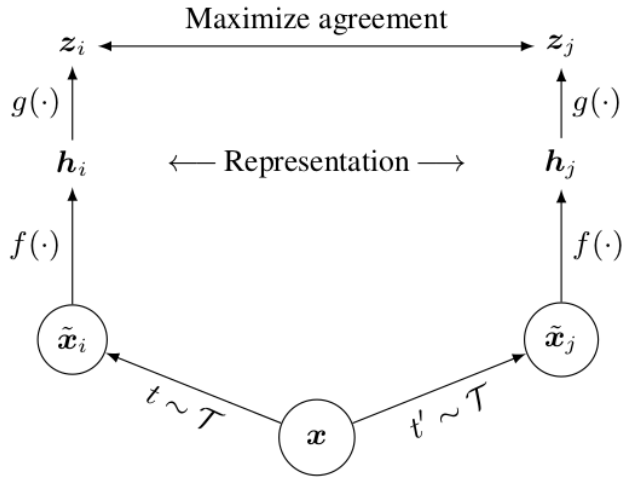


Figure 15: The SimCLR framework. *Source:* [7].

while simultaneously the similarity of $\mathbf{z}_i$ to all other representations in the batch is minimized. If we aggregate those loss terms over the whole batch, we obtain the overall loss as

$$\mathcal{L}_{NT\text{-}Xent} = \frac{1}{2N} \sum_{k=1}^{N} \left[\ell_{2k-1,2k} + \ell_{2k,2k-1} \right]. \tag{8}$$

This formulation assumes that N is the number of original input images in the batch and that the positives pairs are always located at $(2k-1, 2k)$ or $(2k, 2k-1)$ for $k = 1, ..., N$. Also note that $\mathcal{L}_{NT\text{-}Xent}$ is symmetrized by including both $\ell_{2k-1,2k}$ and $\ell_{2k,2k-1}$ as $\ell_{i,j}$ in equation (7) is not symmetric. However, it is sensible that equation (8) is symmetric as the order of random transformations should not have an impact on the loss. Since this loss incorporates the temperature parameter τ and it resembles in its structure the cross-entropy loss, it is called the *normalized temperature-scaled cross entropy loss*, or short NT-Xent [7].

Architecture and Training In [7], a ResNet [18] was used as feature extractor $f(\cdot)$. For the projection head $g(\cdot)$, an ordinary MLP with two layers was chosen. The authors decided to use a projection head at all because this substantially improves the quality of the learned representations. From the theoretical perspective, it would clearly also be possible to directly compute the loss on the feature representations produced by $f(\cdot)$.
Another factor that has a big impact on the performance of this is method is the batch size. It is reported that SimCLR strongly benefits from large

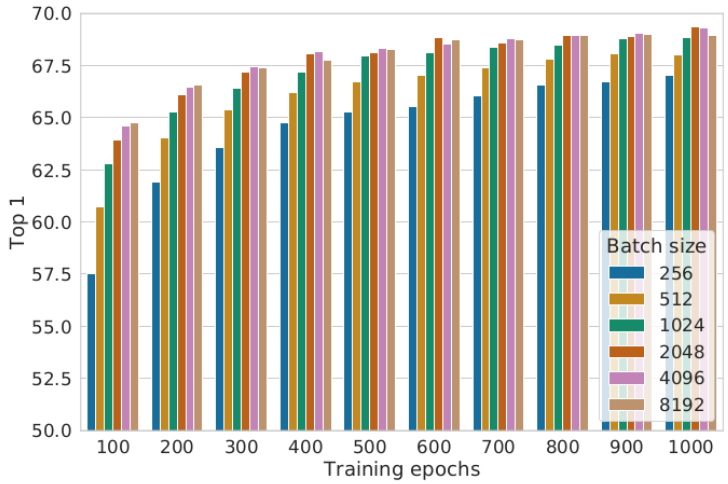


Figure 16: Quality of the SimCLR representations for different batch sizes and training epochs. Performance scores are the top-1 accuracies of a linear classifier trained on top of a frozen feature extractor, which was pretrained with SimCLR. *Source:* [7].

batch sizes. Empirical observations for this relation can be seen in Figure 16. Apparently, both a large batch size and a large number of training epochs are necessary to reach the full potential of SimCLR. Especially, if the number of training epochs is low, the effect of the batch size becomes crucial. The authors reason that this behavior is caused by the fact that large batch sizes and longer training increase the number of negative samples the model sees during training and, therefore, allow to learn better representations. It is also conjectured that unsupervised learning benefits more from larger architectures than supervised settings.

Clustering By now, we have only considered SimCLR as a framework for representation learning. However, in our work, we examined whether it is useful for image clustering. For that, various options are conceivable. The most obvious one is to cluster the learned representations with a clustering algorithm like K-Means. This strategy was pursued in [43, 47] and has shown promising results.
Beyond that, one could also use some modifications like applying a Principle Component Analysis (PCA) on the representations before clustering. Another possibility is to normalize the representations and cluster them with the spherical K-Means algorithm [3]. This has the advantage that the cosine similarity is used as a measure of proximity which resembles the way the representations are processed in SimCLR. In contrast to that, the representations are by design

not guaranteed to work well with euclidean distances.

Discussion A major strength of this method is that it narrows the gap between supervised and unsupervised learning in computer vision. For instance, when training a linear classifier on the representations learned with SimCLR, one can achieve a top-1 accuracy of 76.5% on ImageNet [39]. This is even more remarkable because the framework is relatively simple having only one loss function that is optimized. Furthermore, the fact that the representations are by design trained to be invariant under a chosen set of augmentations is another benefit.
A downside of SimCLR is that it strongly relies on large batch sizes and long training times. Together with large model architectures, this makes this method computationally intense and a single GPU might not suffice to train a network properly.
When it comes to clustering, a drawback is that the representations are not specifically learned for clustering. That is, we have no guarantee that the representations are suitable for clustering as the model is not directly trained for that purpose.

4.3.2 SCAN

In addition to that, we also want to present a method that was specifically developed for image clustering. SCAN (Semantic Clustering by Adopting Nearest neighbors) was proposed in [15] and it consists of three training phases. First, in a pretext task, useful image representations are learned with a framework like SimCLR. Then, a clustering model is trained by aligning the cluster assignments of images and their nearest neighbors in the embedding space. And finally, the clustering network is finetuned with a self-labeling scheme. Subsequently, we will explain these three steps for training SCAN in depth.

Pretext Task In order to obtain a useful prior for the clustering phase, the authors of [15] suggest performing unsupervised representation learning as a pretext task. That is, a network is trained to map input images into an embedding space such that the obtained feature vectors are appropriate for various downstream tasks, for instance, clustering. In doing so, no ground-truth labels are used.
The reason why such a pretraining is beneficial is not only because it reduces the initialization sensitivity of the clustering model, but also that it allows steering the model to focus on the features of interest. This is particularly important for image data as the high dimensionality gives rise to many low-level features we are not interested in or not even aware of. For example, consider a

dataset with a large group of images that all have a certain color in a certain region of the image. Then, it is possible that a clustering network exploits that feature and discriminates samples based on it. Although this may lead to a clear separation of the data, the clustering is probably useless as we are not interested in this kind of random commonalities between images. To avoid that, we can infuse prior knowledge by training with a pretext task. In this case, one can employ contrastive learning to make the representations and as a consequence also the cluster assignments invariant under some transformations. In the aforementioned example, one could destroy such low-level features by augmenting the data with color jittering, translations and rotations. When the network is trained to be invariant under these transformations, the affected features are not captured and can not be exploited in the clustering step.
A concrete example that lets us accomplish this is the representation learning with SimCLR, which we discussed in the previous section. It has precisely the desired property of invariance under certain augmentations and also showed the best performance in clustering with SCAN. But it is stated that various feature learning schemes, like context prediction as proposed in [10] and discussed in Section 3.3, can be chosen in the pretext task. However, it is clearly required that representations of similar images are close to each other in the embedding space.

Semantic Clustering After the pretext task, we can proceed and train a model for semantic clustering. Here, we do not only use the pretrained network for a better initialization but we also collect more information about the clusters by mining nearest neighbors of the representations. That is, we assume that nearest neighbors in the embedding space are likely to belong to the same cluster and train the model to assign the same labels for such pairs of images. By that, we are able to capture more of the variance in the clusters, but we also introduce noise into the learning process.
In the following, let $\Phi(\cdot)$ be a clustering network. Its output is a discrete probability distribution produced by a Softmax layer, i.e. for an image $\mathbf{x} \in \mathcal{D}$, $\Phi_c(\mathbf{x})$ is the estimated probability that $\mathbf{x}$ belongs to cluster $c \in \{1, ..., C\}$. Then, the learning objective to minimize in SCAN is

$$\mathcal{L}_{SCAN} = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \sum_{\mathbf{x}_n \in \mathrm{kNN}(\mathbf{x})} \log\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}_n) \rangle + \lambda \sum_{c=1}^{C} p_c \log p_c \,, \qquad (9)$$

$$\text{where } p_c = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \Phi_c(\mathbf{x}).$$

With $\langle \cdot, \cdot \rangle$, we denote the euclidean inner product. Furthermore, $\mathrm{kNN}(\mathbf{x})$ is the set of images that are the k nearest neighbors of $\mathbf{x}$ in the representation

space of the pretext task. The double sum on the left-hand side enforces that the cluster assignment distributions for neighboring images are similar and, therefore, makes the clustering consistent.
Simultaneously, the term on the right-hand side constitutes an entropy regularization and ensures that the clustering does not degenerate and lead to a single cluster containing all the samples. The hyperparameter $\lambda > 0$ allows us to trade both terms off.
In practice, objective (9) can be easily optimized with mini-batch gradient descent as we only have to replace the whole dataset $\mathcal{D}$ with a batch of samples in the formula.

Self-Labeling To further improve the clustering model, finetuning based on self-labeling is performed in the third training phase. After training the model with objective (9), there exist samples that are assigned to a certain cluster with high confidence, while other samples are rather ambiguous to the model. We assume that examples that are assigned to the same cluster with high confidence are also more likely to indeed belong to the same class. Hence, we can consider these examples as prototypes for the clusters and use their assignments as pseudo-labels. Training with these pseudo labels allows mitigating the effect of the noise introduced by the nearest neighbors when training with the SCAN loss (9). This holds because the k nearest neighbors serve as a kind of preliminary and imprecise self-supervision which is now corrected by putting the focus on confident or "clear" instances.
More concretely, the confident samples are selected as the ones that are predicted to have a probability larger than a threshold τ to belong to a certain cluster. A reasonable value for τ could be 0.99 for example. Then, the network is updated such that the cross-entropy loss of the predicted labels, i.e. the pseudo-labels, and the output for heavily augmented versions of the corresponding confident examples is minimized. In formulas, this means

$$\mathcal{L}_{\textit{self-label}} = -\frac{1}{N_{conf}} \sum_{\mathbf{x} \in \mathcal{D}} \mathbb{1}_{[\max_c \Phi_c(\mathbf{x}) \geq \tau]} \sum_{c=1}^{C} \mathbb{1}_{[c = \operatorname{argmax}_{\tilde{c}} \Phi_{\tilde{c}}(\mathbf{x})]} \log \Phi_c(t(\mathbf{x})), \quad (10)$$

$$\text{where } N_{conf} = |\{\mathbf{x} : \max_c \Phi_c(\mathbf{x}) \geq \tau\}|.$$

Here, N_{conf} is the number of confident examples and $t(\cdot)$ denotes a randomly chosen image augmentation. The random augmentations that are used at this point have to be stronger than the ones in the previous training phases in order to avoid overfitting to the confident samples.
By training to optimize this objective, the network gets more and more confident in its prediction leading to more and more other confident examples

that can be used to generate pseudo-labels. The training is stopped when the number of confident samples N_{conf} does not increase anymore.

Architecture and Training As a feature extractor, a ResNet-18 [18] was used in [15]. On top of that, they added a two-layer MLP as an output head for the representation learning in the pretext task. For clustering and finetuning with pseudo-labels, this head was replaced by a linear classifier as a clustering head. It is also possible to train multiple clustering subheads simultaneously in order to improve the robustness of the model.
The training of the model works as follows: First, the feature extractor is trained on the pretext task. After that, it is used to obtain representations for all the images in the dataset and to mine their nearest neighbors. Then, the backbone of the clustering model is initialized with the feature extractor from the pretext task and the whole network, i.e. the feature extractor and the clustering head, is trained with the SCAN loss (9). When the loss does not decrease anymore, the same network is finetuned with the self-labeling scheme and loss function (10) until the number of confident examples stagnates.

Discussion SCAN is a powerful framework for image clustering and has proven itself successful on common benchmark datasets. For instance, with an accuracy of 88.3% on CIFAR10 [27], SCAN outperformed the state of the art set by IIC [22] by a large margin of 26.6%. This approach is well-thought-out where each of the three training phases has a contribution to the final performance. However, the ablation study in [15] also shows that applying K-means directly on the features obtained by SimCLR already suffices to outperform the state-of-the-art methods, which is quite remarkable and in line with what we report in Section 4.3.1.
Moreover, the enforced invariance under random augmentations is beneficial and leads to truly semantic clusters. We depict some examples as presented in the paper in Figure 17, which demonstrate the high quality of the image clusters.
Another convenience of SCAN is that it appears to be stable under overclustering. That is, when SCAN is trained for a higher number of clusters than there actually are classes in the data, then the accuracy does not drop if the clusters are assigned to superclusters corresponding to the real classes in the evaluation. Interestingly, for the CIFAR100-20 dataset [27], the performance did in fact improve with overclustering. The reason for that is most likely that the intra-class variance can be better explained with multiple clusters than one single cluster in that dataset. All in all, the robustness under overclustering is very desirable as it means that the number of ground-truth classes has not to

Figure 17: Examples for clusters obtained with SCAN on ImageNet-1000. *Source*: [15].

be known in advance. We also make use of overclustering in our experiments (see Section 5.2).
A downside of this approach is its complexity. The fact alone that SCAN requires three training phases makes it computationally expensive but also more prone to factors that have a negative impact on the final outcome. Hence, it may be more difficult to tune all the hyperparameters such that all training phases work well together.

4.3.3 Weak Supervision

Aside from the possibility to use our two-step procedure in an unsupervised way, it gives us also the opportunity to train a segmentation model with weak supervision. This can be realized by providing class labels for the image patches containing the segments. So, the segmentation algorithm extracts the segments in the first step and, in the second step, a classifier is trained on a set of labeled patches. Thereby, we obtain pixelwise image segmentations, although we only need to provide categorical labels for the training patches. As one can easily imagine, creating ground-truth segmentations for training a semantic segmentation model is extremely time-consuming since it is necessary to be very precise and, theoretically, make a decision for every single pixel. In

practice, the process consists of drawing polygons around segments and assigning them to a class. In contrast to that, providing a single class label for an image patch is the easiest possible way of creating a ground truth for an image dataset and takes only a small fraction of the time a proper semantic segmentation requires.
In addition to that, such a weakly supervised approach could be combined with semi-supervised training, i.e. utilizing only a small amount or a small proportion of labeled samples. For instance, one could employ SimCLR for an unsupervised pretraining of a feature extractor CNN and, then, use a few labels to finetune the network and train a classification head.

4.4 Discussion

Now, we want to point out some strengths and weaknesses of our two-step procedure. The reason we eventually came up with the idea of decomposing the task of semantic segmentation was that, in our experiments, the existing methods failed to account for coarse woody debris as its own class (see Section 5.4.2). It seems that these segments are too fine to be detected consistently. Breaking up the problem into two steps mitigates that as having a separate segmentation phase gives us better control over the characteristics of the extracted segments. Then, in the second step, we are still able to capture semantic and high-level features by providing whole patches containing the segments to the clustering algorithm. Thus, we can have very fine segments and meaningful concepts at the same time.
One could also say that our two-step procedure is closer to the way humans perceive objects in images because humans do not analyze every single pixel separately but focus on regions or segments that represent objects.
Furthermore, our method is rather a framework than an architecture, which makes it modular and extendable. For instance, one can choose between different segmentation algorithms and use several clustering models. We also explained how it is possible to infuse weak supervision to improve the performance over an unsupervised setting.
Nonetheless, there are also some issues with this method. One is that it heavily depends on the quality of the segmentation algorithm, i.e. if the segmentation algorithm is not able to extract segments that correspond to the features of interest, this approach can not perform well. This is especially a problem if the features of interest are not as simple as coarse woody debris, which can be identified relatively easily with its straight edges.
Another point to consider in this regard is the tuning of the segmentation algorithm. Even if this is done qualitatively like in our work, this is still some kind of supervisory signal that influences the performance. Although we argue

that the segmentation algorithm can be seen as a preprocessing step, this is a reason why our method is not unsupervised in the purest of ways.
Besides that, our approach might also be computationally inefficient if many segment patches overlap as the overlapping regions are passed into the clustering model multiple times in that case. However, if the size of the patches containing the segments is relatively small and there are on average only few candidate segments per image, the number of pixels fed into the clustering model might even be lower than for the whole original image. For example, if the original images are of the size 256×256 (totaling $256 \times 256 = 65,536$ pixels), then up to 18 patches of size 60×60 (totaling $18 \times 60 \times 60 = 64,800$ pixels) constitute a lower number of pixels to be processed.

5 Experiments

In this section, we provide various experimental results for the methods discussed so far. We start by presenting the dataset for which we developed our approaches. Then, we describe the setting in which we conducted the experiments before we get to their results.

5.1 Dataset

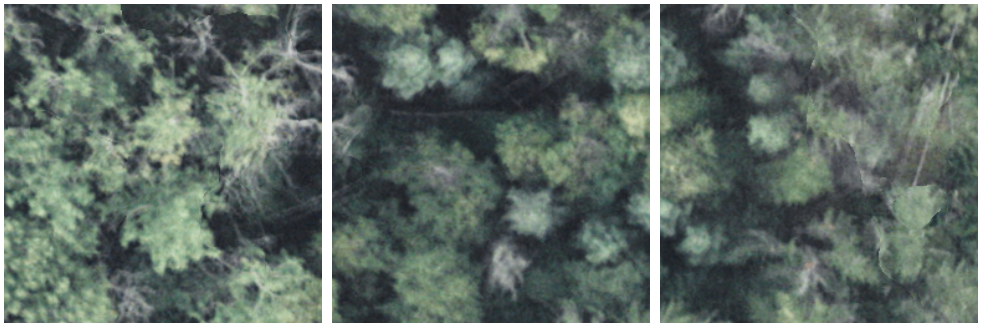
Study Area As already mentioned, the data consist of aerial images of the boreal forest in Alberta, Canada. We depict the exact location where the imagery was recorded in Figure 18. In this region, there are a lot of seismic lines, which can be particularly seen at the example of our study area in Figure 21.

Figure 18: Location of the study area (blue pin). *Source: ©OpenStreetMap contributors*[1].

The so-called *ground sampling distance* (GSD) of the images is 5cm. That means that one pixel covers an area of 5cm by 5cm, which is a comparably high resolution for aerial images.
Moreover, the images of the study area were taken in two different seasons, namely spring and summer. The reason for that is that, at the point in time in spring, deciduous trees were not covered with leaves, whereas this was the case at the second point in time in summer. A comparison of the seasons can be seen in Figure 19. The consequence with respect to the detection of coarse

[1] `https://www.openstreetmap.org/#map=14/55.3748/-111.1477`

(a) Spring



(b) Summer

Figure 19: Three sample images in the two different seasons. The rows correspond to the exact same locations.

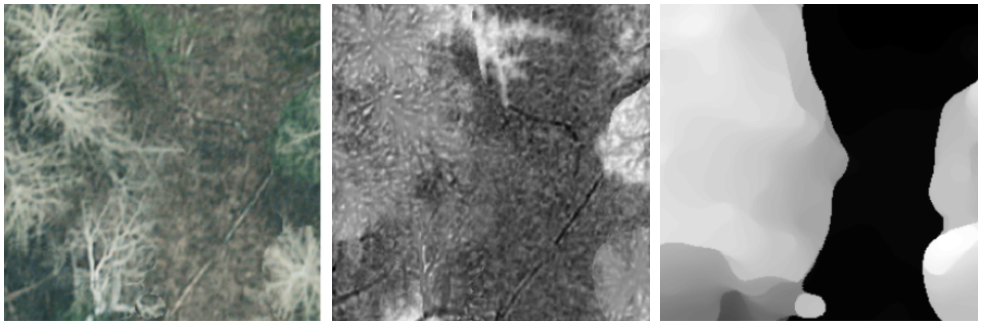
woody debris is twofold: On the one hand, if many trees are not covered with leaves, the view on the ground and the logs lying on it is better. But on the other hand, distinguishing between branches of living trees without leaves and coarse woody debris gets harder as both objects have similar visual features. This was one of the main error sources for our models (see Figure 24 (b)).

Input Bands The data do not only contain the standard image channels Red, Green and Blue but also two others. The first is a near-infrared (NIR) band. It displays wavelengths that are longer than those of visible light but still at the lower end of the infrared spectrum. The NIR channel can be used together with the RGB values to obtain the so-called *Normalized Difference Vegetation Index* (NDVI) [9]. It is defined as follows:

$$NDVI = \frac{NIR - RED}{NIR + RED} .$$

That is, the values are always in the interval $[-1, 1]$. The index can be seen as a measure of how healthy vegetation is. This is particularly useful for the detec-

tion of coarse woody debris because dead organic material is accompanied by low NDVI values, whereas living trees have a higher index (see Figure 20 (b)). The other additional input channel is a *Digital Surface Model* (DSM). It is derived from LIDAR point clouds and indicates the elevation of the tallest object at a certain location. Unfortunately, a *Digital Terrain Model* (DTM) indicating the elevation of the ground was not available. This would have allowed the creation of a *Canopy Height Model* (CHM). A CHM is calculated as $CHM = DSM - DTM$ and, therefore, specifies the actual height of the trees and objects in the study area [44]. This might have been advantageous as the absolute elevation of a DSM is irrelevant for our task and it is already filtered out in a CHM. An example of how these bands look in practice can be seen in Figure 20.



(a) RGB (b) NDVI (c) DSM

Figure 20: Different bands of the imagery.

Preprocessing The dataset in its original form consisted of large GeoTIFF-files. So, not only the band values are provided for each pixel, but the imagery is also georeferenced, i.e. geographic coordinates can be extracted for every pixel. This is important when it comes to applying the results of the analysis, but for the modeling itself, the coordinates are not used.
To prepare the data for model training, the files had to be sliced into tiles of a suitable format. For this, we utilized QGIS [37], an open-source geoinformation system. The tiles were chosen to have a size of 256×256 pixels, which is in the range of image sizes in common datasets. Furthermore, we did not use the whole entirety of the data for training. Since the dataset is extremely large and also quite homogeneous, we decided to extract the tiles only from a rather small subset of the study area (see Figure 21). This gives us still sufficient data to learn from. More concretely, it resulted in $13,886$ training images in the format as shown in Figures 19 and 20.

Aside from that, all values were scaled to be in the interval $[0, 1]$. Additionally, we shifted the DSM values for every image such that the minimum was zero. In doing so, we hoped that we could create a pseudo-version of a CHM such that the models focus rather on differences in the DSM values than on the absolute elevations. Especially in the unsupervised setting where we can hardly control which features the clustering is based on, we assume this shifting to be a sensible preprocessing step.

Validation and Test Data At first, no test data containing ground-truth annotations were provided. So, the validation of prototypes has been solely qualitative for large parts of this work. However, this is problematic for comparing models which is why we manually created evaluation datasets. We did this by drawing polygons around the instances of coarse woody debris in the QGIS environment. In the end, we annotated 100 images for validation and 150 images for testing. Compared to the size of the training dataset, this is quite small, but the pixelwise evaluation and the homogeneity of the data compensate for that. An overview of the different areas for training, validation and testing is given in Figure 21.

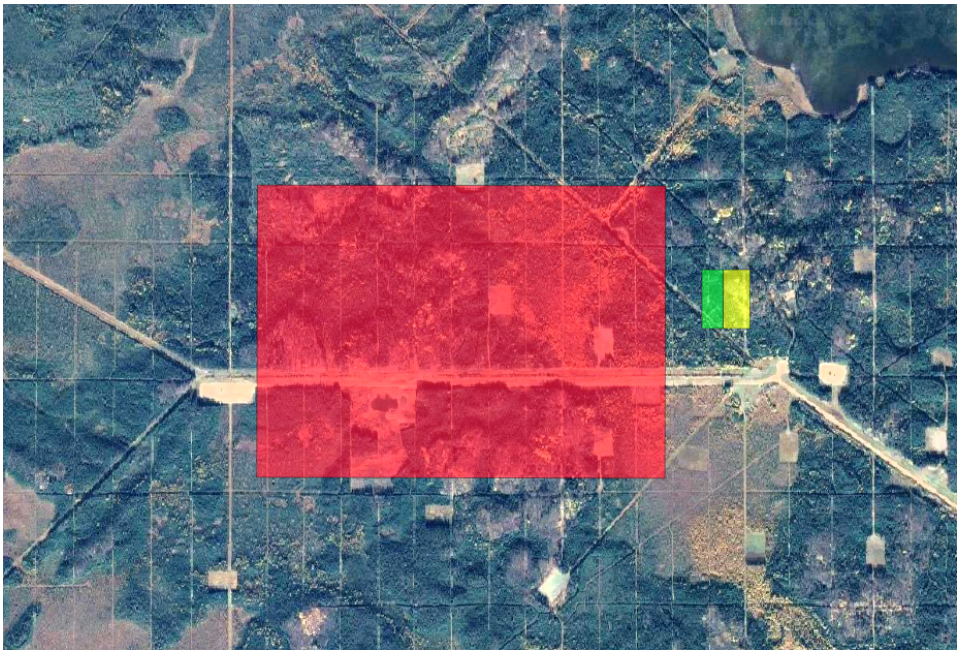


Figure 21: Areas used for training (red), validation (green) and testing (yellow).

5.2 General Setting

Performance Metrics As our problem is framed as semantic image segmentation with binary labels (coarse woody debris vs. rest), we can use standard evaluation metrics for the assessment of the trained models. We employed the F1-measure, the Jaccard index as well as precision and recall on pixel level. That is, we assigned each pixel to one of the categories *True Positive*, *True Negative*, *False Positive* and *False Negative* and aggregated these to the mentioned scores. Here, the pixels belonging to segments of coarse woody debris are considered as the real positives. Also note that, in the context of image segmentation, the pixelwise F1-measure is commonly dubbed *Dice score*, whereas the Jaccard index is usually referred to as *Intersection over Union* (IoU). The latter makes sense in a way that it, in fact, measures the area where both the ground truth or the predictions indicate coarse woody debris and divides it by the area where at least one of ground truth and prediction does so. Hence, the IoU is easily interpretable.
We abstain from using the accuracy as a performance metric like it is done in many other works because it is misleading for our task. The reason for that is that the dataset is extremely imbalanced and coarse woody debris accounts for less than 0.4% of the test area. Thus, a trivial model predicting no coarse woody debris at all would yield an almost perfect accuracy despite being completely useless.

Overclustering By definition, unsupervised models have no supervisory signal that gives them an intuition on which features could or should be used to achieve the desired classification. Consequently, the obtained clusters can represent different concepts than actually expected. Especially for our task, this is problematic. As we stated above, coarse woody debris covers only a tiny fraction of the area in the images. Therefore, an unsupervised binary segmentation model is very unlikely to produce the desired categorization as other features are much more evident. For instance, it is much more likely that a binary segmentation separates ground from standing trees if trained without supervision.
To overcome this issue, we used overclustering and oversegmentation, i.e. we trained the models to predict more than the two desired labels. Thereby, we hope that not only the obvious and evident features but also minor ones like coarse woody debris are captured. If not specified otherwise, we use $k = 16$ as the number of artificial labels in the following experiments. We analyze the effect of overclustering in Section 5.5.5.

Mapping Clusters to Classes As a consequence of the described overclustering, we are faced with the problem of determining which clusters really represent coarse woody debris and which ones do not. To this end, we implemented a step-wise selection scheme, which we applied on the validation data. That is, we maintained a set of labels (initially empty) that correspond to coarse woody debris and, in each step, added or removed one label from the set that leads to the largest improvement on the validation set. When the set can not be improved anymore, we have reached the best mapping from labels to the real two classes.
Moreover, the validation set was used for hyperparameter tuning. In most of the cases, we conducted manual tuning as we could combine this with a qualitative visual inspection of the results. However, one has to be aware that the scores on the validation set are not comparable if different numbers of labels were used for overclustering. This holds because a higher number of labels leads to a better separation of the data the mapping was determined on, but also generalizes worse on unseen test images.

5.3 Implementation and Training Details

The code for this project is based on the deep learning framework PyTorch [35]. To accelerate training, we worked with an NVIDIA TITAN X (Pascal) GPU with 12 GB of memory. Furthermore, we trained all our models with the Adam optimizer [26]. In the following, we provide specific details on the implementation and the training process for our different methods.

5.3.1 Baselines

K-Means As the most basic approach, we performed K-Means clustering on the pixel features. We did not only use the five input bands as described in Section 5.1, but also included an additional channel indicating edge activations. More precisely, we applied the filter of Section 4.2.2 on the RGB image together with the NDVI band and scaled the feature maps in each batch to the interval $[0, 1]$ afterward. For the clustering itself, we used mini-batch K-Means and chose 5e5 as the batch size.

Segmentation via Invariant Information Clustering (IIC) For semantic image segmentation as introduced in Section 3.1, we employed the authors' official implementation[2]. As random image perturbations, we applied color jittering as well as the geometric transformations rotation, flipping and shearing.

[2] `https://github.com/xu-ji/IIC`

Since it is suggested in [22], we also added edge features of the Sobel operator [41] to the other input channels. As network architecture, a VGG11-like backbone and a segmentation head consisting of a 1×1-convolution and a Softmax layer were used. We trained the model with a batch size of 10 for two rounds where in each round the overclustering head was trained for one epoch and the main output head was trained for two epochs. The learning rate was initially set to 1e$-$4 and decayed by the factor 0.1 after the first training round. For the displacements (t_1, t_2) in equation (3), we set the upper bound to one pixel. The reason for that is that we deliberately wanted to keep the spatial continuity as low as possible such that thin logs do hopefully not get smoothed out.

W-Net We implemented the W-Net ourselves. In most parts of the architecture, we exactly followed the description of [45], e.g. we used the suggested depthwise-separable convolutions. However, we also integrated two minor amendments. Firstly, we extended the contracting and the expanding path of both U-Nets by one block each. And secondly, to account for the resulting computational overhead, we halved the number of channels in each block.
Furthermore, we modified the reconstruction loss in a way that it focuses more on the features that are interesting for us. We did this by utilizing a weighted L2 loss instead of its standard version. The weight for each pixel was chosen as the Sobel activation of the input image at that pixel. In doing so, we forced the network to segment and reconstruct more precisely on edges, which is where coarse woody debris is mostly at. To ensure that the reconstruction and the soft-N-cut loss are in the same order of magnitude and to relax the constraints on spatial continuity of segments, we used the convex combination $\mathcal{L} = 0.001 \cdot \mathcal{L}_{soft\text{-}N\text{-}cut} + 0.999 \cdot \mathcal{L}_{reconstr}$ as total loss. Note that the balancing parameter needed to be that large for $\mathcal{L}_{reconstr}$ to account for the mostly small Sobel weights. Our implementation of $\mathcal{L}_{soft\text{-}N\text{-}cut}$ is based on this code[3]. For the parameters σ_X^2, σ_I^2 and r, we chose the values 1.0, 10.0 and 5.0.
To even further relax the spatial continuity, we did also not apply the proposed postprocessing steps as these lead to smoother and larger segments. This may be desirable in general, but for our problem, it degrades the ability of the model to detect such fine segments as thin logs. When training the W-Net with a batch size of 10, a dropout of 0.65 and a learning rate of 0.01, the network converged after only one epoch. The number of oversegmentation labels was set to $k = 8$ here.

[3] `https://github.com/fkodom/wnet-unsupervised-image-segmentation/blob/master/src/loss.py`

Unsupervised Visual Representation Learning by Context Prediction To employ the representation learning framework of [10] for image segmentation, we pursued the approach described in [22]. However, their implementation[4] has certain issues. First of all, their prediction head produces nine logit values for one pair of image patches. But there are only eight possible spatial relations (see Figure 7) which is why this is an implementation flaw. On top of that, their CNN used for dense feature extraction only contains one pooling layer which is quite few.
Therefore, we wrote our own implementation of this method inspired by the proposed one. As a backbone CNN, we chose a combination of three blocks where each block consists of two convolutions, two batch normalization layers and two ReLU activations. The blocks are connected with max-pooling layers and the last block is followed by a 1×1-convolution to transform the representations to the desired dimensionality. To reconstruct the spatial resolution of the input, we applied nearest-neighbor upsampling. That is, the introduction of a second pooling layer comes at the cost of losing spatial precision. For the context prediction, we added an MLP with two hidden layers producing scores for the eight possible relative positions.
As proposed in [10], we applied color dropping, i.e. we replaced two randomly chosen color channels with Gaussian noise in order to prevent trivial solutions due to chromatic aberration (see Section 3.3). Furthermore, we did not use the DSM as an input channel for this method because we noticed that this heavily impacted the results in a negative way.
When learning the representations, we chose to train for four epochs with 1152 image patches of 36×36 pixels per batch. As a loss function, the cross-entropy loss was employed. The learning rate was $1e{-}3$ and the dimension of the representations was 32, which is rather low because we extract those representations densely for each pixel at test time. With this setting, the accuracy in the context prediction task went up to about 55% during training, meaning that the prediction was about four times better than random guessing.
To obtain segmentation maps from the learned representations, we applied mini-batch K-Means clustering with $k = 8$ possible clusters. In Table 2, we refer to this method as *Doersch + K-Means*.

5.3.2 Our Two-Step Procedure

Segmentation In the segmentation part of our proposed two-step procedure for semantic segmentation, we used the algorithm introduced in Section 4.2.2. For the extraction of straight edges, we aggregated the input channels via $RED + GREEN + BLUE - 3 \cdot NDVI$. The negative sign in front of the

[4] `https://github.com/xu-ji/IIC`

NDVI channel is sensible because coarse woody debris is often accompanied by both bright grayish color and a low NDVI. The resulting feature maps were binarized by thresholding at a value of 10.0. The DBSCAN clustering was performed with the parameters $\epsilon = 1$ and $min\_samples = 2$.
To speed up the retrieval of the segments for clustering, we ran the segmentation once for every image and stored patches of 60×60 pixels containing the segments on disk. Therefore, we could leave out the whole segmentation step during the training of the clustering models.

Clustering with IIC For the clustering of segment patches with IIC, we created our own implementation to have better control and monitoring during the process. As a feature extractor, we chose a ResNet-18 which was followed by a two-layer MLP to produce the logits. Our hyperparameter setup consisted of a batch size of 512, eight training rounds and a learning rate of 1e−4. As for the segmentation with IIC, each training round contained one epoch for training the overclustering head and two epochs for the main head. After the fourth and seventh round, we decayed the learning rate with a factor of 0.1. As data augmentations, we used color jittering, rotations, flipping, small translations, adding noise and also blacking out regions at the borders of some images.

Clustering of Representations Learned with SimCLR To learn representations with SimCLR, we plugged the loss and the training functions of this repository[5] into our code. We trained a ResNet-18 for 200 epochs with a batch size of 1024 and a weight decay of 1e−4. That is the dimension of the representations was 512. The temperature for the NT-Xent loss in equation (8) was 0.5. During the first five epochs, we linearly ramped up the learning rate to 0.02 and performed cosine annealing [29] after this warm-up phase. Moreover, we applied the same image transformations as for the clustering with IIC.
In order to obtain a clustering based on these representations, we used two different settings. The first one was to perform a PCA to reduce the dimensionality of the features as suggested in [47]. Then, the ordinary K-Means algorithm was trained on the transformed representations.
The second approach was to normalize the representations such that they lie on the unit sphere and apply spherical K-Means clustering[6]. That is, cosine similarities are used as a proximity measure instead of the euclidean distances. We dub these two models *SimCLR + PCA + K-Means* and *SimCLR + spherical K-Means* in Table 2.

[5] `https://github.com/wvangansbeke/Unsupervised-Classification`
[6] `https://github.com/jasonlaska/spherecluster`

Clustering with SCAN For the image clustering with SCAN, we employed the necessary functions and utilities from the authors' codebase[7]. The SimCLR pretext was solved with the same configuration as for the representation clustering. In the second phase, the semantic clustering, we set the number of epochs to 20 and the batch size to 512. We trained only one subhead to ensure a fair comparison. The learning rate and the weight decay were both 1e−4. Furthermore, we used 50 nearest neighbors to augment the clustering and we chose 2.0 as the entropy weight in equation (9).
In the self-labeling step, we trained for ten epochs with a batch size of 512, a learning rate of 1e−5 and a weight decay of 1e−4. A prediction was considered confident if the estimated probability was larger than 0.99. We also balanced the different labels with class weights during training. As it is suggested to use stronger augmentations for this training phase in [15], we intensified the noise and the color jittering compared to the pretext and the SCAN phase. Nonetheless, we noticed that the training via self-labeling was prone to overfitting long before the fraction of confident examples saturated. Therefore, a careful observation of the performance on the validation set was necessary.

5.4 Results

5.4.1 Quantitative

In Table 2, we list the performance scores of the discussed approaches for different metrics.

Method	Dice	IoU	Precision	Recall
Baselines:				
K-Means	10.3 ± 2.2	5.4 ± 1.2	6.1 ± 1.7	39.7 ± 7.4
IIC Segmentation	2.6 ± 0.4	1.3 ± 0.2	1.3 ± 0.2	47.5 ± 13.1
W-Net	1.9 ± 1.7	1.0 ± 0.9	1.1 ± 1.0	69.0 ± 26.7
Doersch + K-Means	1.6 ± 0.7	0.8 ± 0.4	0.8 ± 0.4	35.1 ± 10.3
Two-Step Procedure:				
IIC	21.2 ± 0.8	11.9 ± 0.5	13.3 ± 0.6	53.1 ± 0.3
SimCLR + PCA + K-Means	29.9 ± 0.3	17.6 ± 0.2	20.1 ± 0.1	58.0 ± 1.6
SimCLR + spherical K-Means	29.1 ± 0.4	17.0 ± 0.3	20.1 ± 0.3	52.8 ± 1.1
SCAN	30.3 ± 2.5	17.9 ± 1.7	22.0 ± 2.2	48.8 ± 3.1

Table 2: Performance scores of the presented methods. We provide the average over five runs together with the corresponding standard error. For all methods and all runs relying on a pretrained SimCLR model, the same backbone was used to reduce computational efforts and ensure comparability.

[7] `https://github.com/wvangansbeke/Unsupervised-Classification`

Apparently, SCAN achieves the best performance with a Dice score of 30.3 and an IoU of 17.9. Although being conceptually much simpler, the combination of SimCLR, PCA and K-Means performs similarly to SCAN in our experiments. It is also able to outperform IIC, which is in line with the findings of [15].
The overall conclusions we can draw from these results are twofold. One can see that all of the baseline models perform extremely badly on our dataset. However, all of our proposed two-step procedures are able to outperform the baselines by a large margin. Hence, we can say that our method is justified by this major improvement in performance.
On the other hand, all of the scores are rather low which makes it questionable whether this task is solved in a satisfactory manner at all. To answer this, we firstly refer to the qualitative analysis of models and, secondly, we argue that the pixelwise metrics are very pessimistic for the dataset. The reason for that is that coarse woody debris forms thin and long segments most of the time. Thus, a large fraction of the pixels lies on or close to the border of the coarse woody debris segments. And as one can easily imagine, unambiguously assigning a border pixel to a class can be very difficult for a model as well as for a person creating ground-truth labels. In a pixelwise evaluation, this circumstance is not taken into consideration which is why these metrics tend to produce pessimistic scores. We give an example of this behavior in Figure 22.

(a) Image (b) Ground truth (c) Prediction

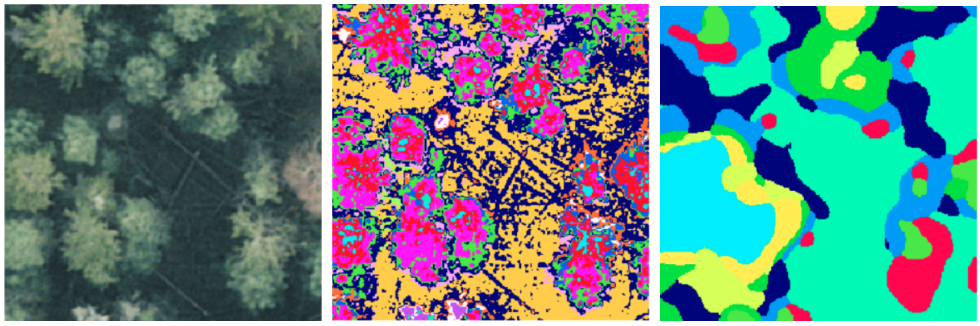
Figure 22: An image, its ground-truth segmentation and an exemplary corresponding prediction. Although almost all instances of coarse woody debris are detected and the prediction looks really good from a qualitative perspective, the resulting performance scores being a Dice coefficient of 49.7%, an IoU of 33.1%, a precision of 39.9% and a recall of 66.1% are unexpectedly low.

As we already explained in Section 2.1, the evaluation in [28] does not suffer from this because their GEOBIA approach assumes that the boundaries of the ground-truth segments are perfectly aligned with those of their segmentation. We also mentioned that, therefore, the scores of their and our work are not

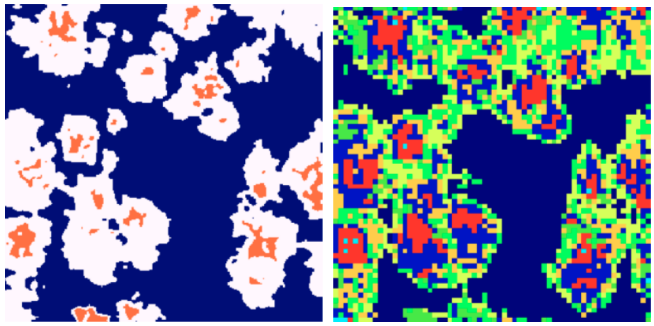
directly comparable, but it seems that our unsupervised approaches are not able to rival their supervised model.

5.4.2 Qualitative

Nevertheless, to not only showcase the quantitative improvements of our work over existing unsupervised methods, we conduct a qualitative analysis of the utilized models. This also demonstrates the value of our work for the application in the real world.



(a) Original (b) K-Means (c) IIC Segmentation



(d) W-Net (e) Doersch + K-Means

Figure 23: Example image and oversegmentations created with different baseline models. We show the oversegmentations instead of the binary segmentations after selecting the labels for coarse woody debris because it provides more insights about the methods.

K-Means Since K-Means clustering on pixel level was not primarily designed for image segmentation, it shows some weaknesses in our task. As we illustrate in Figure 23 (b), the segments are extremely fragmented and many tiny, spurious segments exist. On top of that, the segment containing the logs

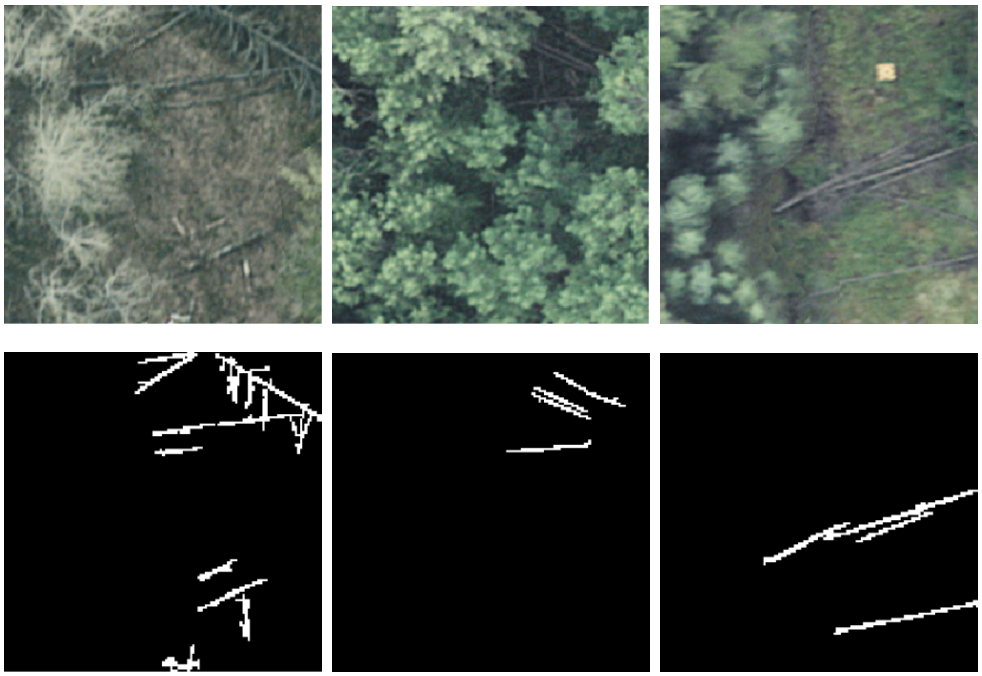
(dark blue) also contains many other pixels. Nonetheless, at least in this example, the model seems to have learned and separated some features that can be associated with coarse woody debris which is already an advantage over the other baselines.

IIC Segmentation The oversegmentation in Figure 23 (c), depicts an example of a prediction produced with IIC for segmentation. Unfortunately, the model is not able to separate the logs in the center from the ground, although we set the hyperparameters such that small segments like the red ones are allowed. Compared to the others, this oversegmentation looks still quite smooth.

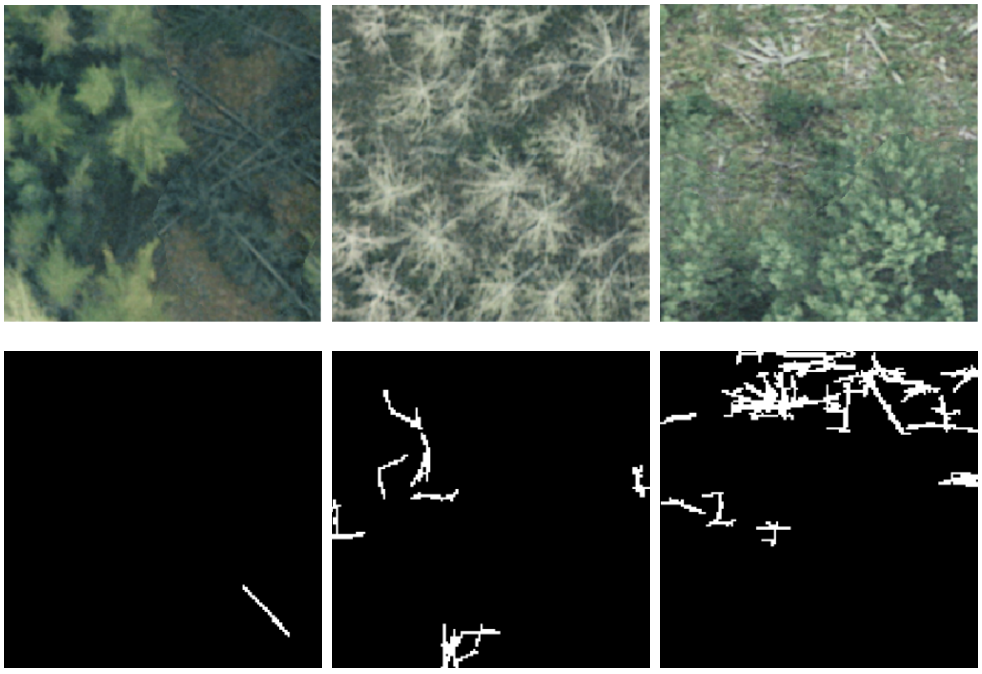
W-Net We make similar observations for the W-Net for which we provide an example in Figure 23 (d). Apparently, the logs in the center of the image are too vague to be captured as their own segment and, therefore, simply treated as part of the ground segment. Even though we modified the reconstruction loss and chose the hyperparameters to relax the spatial continuity, we could not resolve this issue.
On the other hand, we can not say that the segmentations of the W-Net, like the one in the example, are completely useless. It is clear that the predicted segments are related to the image in a meaningful way. The problem is rather that the features of interest for us are not detected, while, for instance, trees are identified reasonably well.

Doersch + K-Means We show an example for this segmentation method in Figure 23 (e). The oversegmented image looks very fragmented at the first glance, but, once again, we can see that the segments represent objects or parts of objects in the original image. Beyond that, this model produces segmentations at a lower resolution than the others.

Our Two-Step Procedure As all of the different variants we have tried for our two-step procedure rely heavily on the segmentation algorithm, the results differ only in the quality and purity of the clustering. Thus, they look in principle quite similar and we only give examples for segmentations obtained with SCAN for demonstration purposes here (see Figure 24).
The good examples show that, with this method, we are able to capture fine segments and assign them to the class of coarse woody debris consistently. It is remarkable that even very subliminal and badly visible features, like in the second image, are detected.
Of course, there are also some situations where this method performs less well. In the first image of the failure examples, multiple logs on the ground are not



(a) Good examples



(b) Failure examples

Figure 24: Good and bad examples for the segmentation of coarse woody debris with our two-step procedure and SCAN clustering.

detected as they are possibly too dark to be recognized as such. The second sample illustrates that branches of trees without leaves are mistaken for coarse woody debris on some occasions. Furthermore, we can see a weakness of the segmentation algorithm in the third image. It extracts several small logs or snags at the top of the image, but it is not able to separate them from each other while overestimating the area covered by them. Hence, the prediction contains a rather large area of coarse woody debris although there are only relatively small instances of it.
Nevertheless, the overall quality of these predictions is by far better than the ones we have observed for the baseline models, which underlines the advantage of splitting the task of semantic segmentation into two separate steps.
In Appendix A, we visualize some more segmentation examples produced with our method. We also show a small selection of image clusters that were obtained with SCAN.

5.5 Further Experiments

5.5.1 Ablation Study

As our proposed framework consists of two steps, the question arises how much both of these contribute to the final performance. For the first, namely the segmentation step, it is hard to perform a proper ablation since it is unclear to which entities we should assign semantic labels in the second step if we do not extract segments beforehand. However, one could argue that our experiments with the baseline models (see Table 2) already constitute an ablation study for the segmentation step because, for these methods, semantic labels are predicted for pixels directly instead of entire segments.
On the other hand, the semantic label assignment step is, in theory, easily dispensible. We can simply assign the same label to all of the extracted candidate segments and assess the resulting performance. When we did this, i.e. when we treated every segment produced by the segmentation function described in Section 4.2.2 as coarse woody debris, we observed a Dice score of 10.7% and an IoU of 5.6%. Moreover, the recall, with a value of 66.7%, was comparably high, whereas the precision amounted to only 5.8%. This is quite obvious because the initial set of candidates is not refined by a clustering algorithm if we omit the second phase.
Remarkably, the overall performance seems to be in the range of the segmentation with K-Means, which we presented as a baseline model (see Table 2). This emphasizes once again that our specifically designed segmentation algorithm is a relatively strong prior despite not containing any learned parameters.

5.5.2 Employment of Weak Supervision

In Section 4.3.3, we mentioned that it is possible to leverage our two-step procedure to infuse weak supervision during training. We pursue this strategy here to determine what amount of supervision is necessary to beat the unsupervised models and to obtain another comparison for them.
For the experiment, we provided binary labels for 400 segment patches produced by our segmentation algorithm of Section 4.2.2. These were utilized to train a classification network. In particular, we used a ResNet-18 backbone with a two-layer MLP on top of it and trained for 80 epochs with batches of size 32. The learning rate was 1e−4 for the first 60 epochs and decayed once by the factor 0.1 after that. The label distribution was imbalanced which is why we used a weight of 1.5 for the class containing coarse woody debris in the cross-entropy loss. To prevent overfitting to the small amount of data, we used a weight decay of 0.05. In addition to that, we augmented the labeled patches by rotations, flipping and blacking out border regions.
With this setting, we trained the network with different amounts of labeled samples and examined the performance. The results are provided in Figure 25.

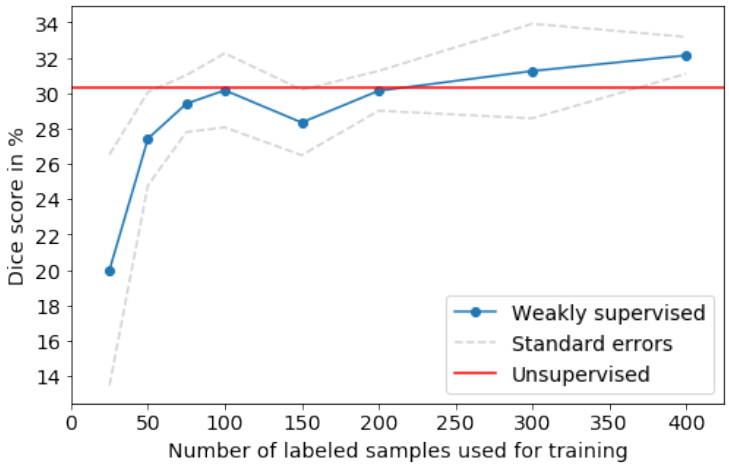


Figure 25: Performance of a weakly supervised model when training with different amounts of labeled samples. Scores are averaged over five runs.

As we can see, 100 labeled samples suffice to rival the performance of the unsupervised SCAN model of Section 5.4. On the other hand, the fact that the weakly supervised model is not able to clearly outperform the unsupervised model with up to 300 labels speaks in favor of the quality of the latter. But also note that the segmentation algorithm in the first step becomes the performance bottleneck when increasing the number of labeled samples. We depict some qualitative examples for the weakly supervised approach in Appendix A.

5.5.3 Influence of Input Channels

We also assessed, to which extent the individual input bands have an impact on the performance. In doing so, we trained a feature extractor with SimCLR on image patches that had different combinations of the available channels. That is, we trained the model for the channel combinations RGB, $RGB + DSM$, $RGB + NDVI$ and, of course, the full data with $RGB + DSM + NDVI$. After that, we applied a PCA on the resulting representations and performed five runs of K-Means clustering. We employ this method for the semantic label assignment here because it is both faster and more robust than SCAN during training despite showing a slightly worse performance (see Table 2). We also take this method as a representative for our two-step procedure in the following experiments for the same reasons. When composing these clustering models with our segmentation algorithm of Section 4.2.2, the two-step procedure led to the scores depicted in Figure 26.
It is important to note that not only the clustering of the segments but also the segment extraction depends on the input channels. That is, for the settings RGB and $RGB + DSM$, the NDVI band was not used for the segmentation step. Thus, the channel aggregation before the detection of straight lines was chosen to be the mean over the RGB channels (instead of $RED + GREEN + BLUE - 3 \cdot NDVI$) and the threshold for binarization was set to 5.0 (instead of 10.0). In this regard, one should also bear in mind that the quality of the extracted segment candidates is highly sensitive to the hyperparameters.

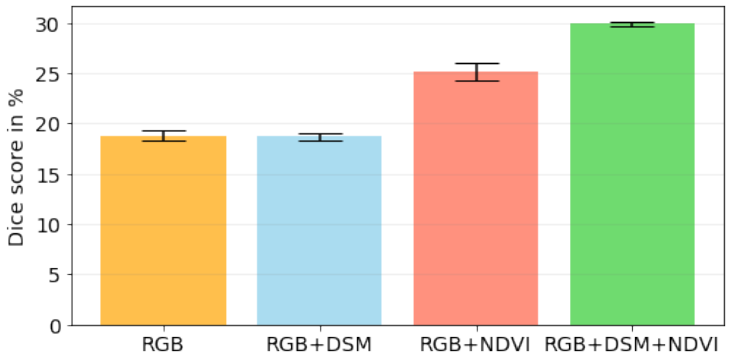


Figure 26: Dice scores for models trained on different input bands. Scores are averaged over five runs and error bars indicate the corresponding standard deviations. The feature extractor trained with SimCLR was fixed per setting across the runs to reduce the computational cost.

Apparently, the NDVI has a positive influence on the detection of coarse woody debris. This is in line with [28] where it is stated that the NDVI is the most

important predictor variable. The best performance is achieved when the full data is used. Aside from that, we observe that the inclusion of the DSM seems to improve the model only in combination with the NDVI.

5.5.4 Transferability between Seasons

In Section 5.1, we mentioned that our imagery was recorded in two different seasons, spring and summer. For the practical application, it is interesting how well a model trained on one season generalizes to another. To analyze that, we trained our *SimCLR + PCA + K-Means* model on the full data, the spring data and the summer data separately. Then, we evaluated the models separately for the two seasons. We report the results of this experiment in Figure 27.

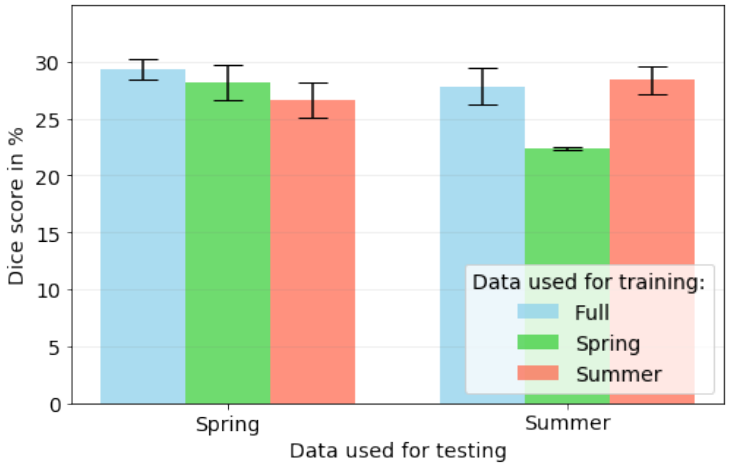


Figure 27: Model performances on the two seasons when training with different data. The averaged scores and standard errors were obtained from five runs with a fixed SimCLR backbone for each setting.

This diagram tells us that the model trained on the full data performs well on both seasons, which is what one would expect. However, we can see that the model trained on the imagery recorded in spring generalizes much worse to summer than the other way around. Therefore, in a scenario where one can only acquire training images in one season but wants to do inference for multiple points in time, the summer season might be a sensible choice for the training data due to the resulting model robustness.
Yet, one has to keep in mind that the data from the different seasons also exhibit different amounts of coarse woody debris. While, in spring, coarse woody debris covers about 0.46% of the test area according to the ground-truth segmentation, the corresponding value for the summer data amounts

to only 0.26%. Since both data come from the exact same area, we assume that the real difference is not that large. So, this speaks in favor of using the imagery from spring for inference.

5.5.5 Effect of Overclustering

Since coarse woody debris is not the most obvious feature in our imagery, we have to perform overclustering during training and determine the set of clusters corresponding to coarse woody debris afterward (see Section 5.2). Thereby, it is not clear how many clusters we should use during training and how this affects the final performance of the models. Thus, we assessed the sensitivity with respect to overclustering in an experiment. The result can be seen in Figure 28.

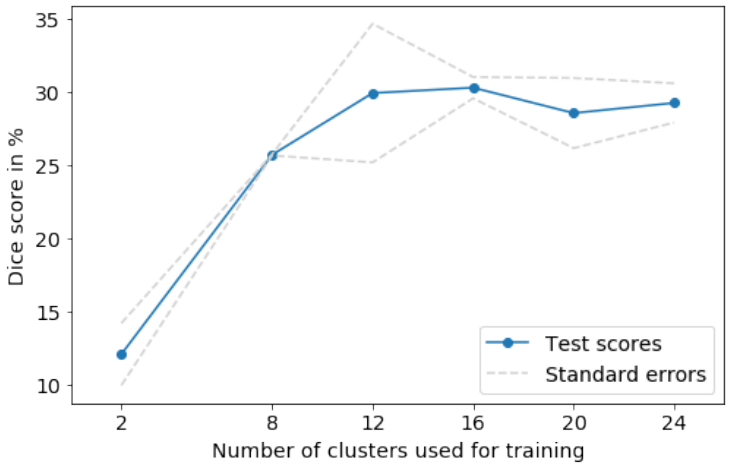


Figure 28: Performance scores obtained when training with different numbers of clusters. Once again, we averaged the scores of five runs with the proposed *SimCLR + PCA + K-Means* method and used the same feature extractor for each setting.

Clearly, the number of ground-truth clusters, i.e. two, is not suitable. This supports our assumption that other more prominent features are exploited for clustering in that case. The maximal scores were obtained at 12 and 16 which justifies our choice of 16 clusters in the experiments. If the number of clusters is further increased, a very slight tendency to overfitting in the cluster selection can be surmised.

6 Conclusion

In this work, we studied the problem of unsupervised semantic segmentation of coarse woody debris in aerial imagery. We discussed several state-of-the-art methods for unsupervised image segmentation and showed that they are not capable of detecting small and low-key features such as coarse woody debris.
Therefore, we developed a novel framework for semantic segmentation that mitigates this issue. In our two-step procedure, segments are extracted as image regions by a non-semantic segmentation algorithm in the first step. In the second step, we train models to cluster image patches containing the extracted segments. Since we separate the segmentation step from the semantic label assignment, it is easier to define the characteristics of the segmentation and to ensure that the features of interest are captured properly. It also allows us to employ approaches from the field of image clustering which provides more literature than the field of unsupervised semantic segmentation at this point in time.
With the proposed concept, we manage to outperform existing methods for unsupervised segmentation by a large margin. However, as an unsupervised learning framework, it is still far from perfect and not able to rival supervised approaches. Nevertheless, our method is able to advance to the regime of useful predictions, whereas the existing unsupervised approaches clearly fail to do so.
A limitation of our two-step procedure can be found in the segmentation step. For our problem, we specifically designed a rule-based segmentation algorithm that was suitable. But in general, this might be a conceptual bottleneck as the overall performance heavily depends on the quality of the extracted segments. As future work, it would be interesting to integrate the parts of our two-step procedure into an end-to-end trainable model. Yet, this is not easily achievable as the forming of segments has to remain uncoupled from the semantic label assignment. Aside from that, there might be space for improvement in the way the segments are represented in the clustering phase. Some more sophisticated formats than simple image patches for the segments, e.g. a combination of patches and masks, could possibly facilitate finding the desired clusters and strengthen the performance.

References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[2] Adam Bielski and Paolo Favaro. Emergence of object segmentation in perturbed generative models, 2019.

[3] Christian Buchta, Martin Kober, Ingo Feinerer, and Kurt Hornik. Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22, 2012.

[4] Holger Caesar, Jasper R. R. Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. *CoRR*, abs/1612.03716, 2016.

[5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016.

[9] DLR - Earth Observation Center. Normalized Difference Vegetation Index (NDVI). `https://www.dlr.de/eoc/de/desktopdefault.aspx/tabid-9142/19518_read-45426/`. Accessed: 2020-09-21.

[10] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015.

[11] Fuzhou Duan, Yangchun Wan, and Lei Deng. A novel approach for coarse-to-fine windthrown tree extraction based on unmanned aerial vehicle images. *Remote Sensing*, 9(4):306, Mar 2017.

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[13] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 09 2004.

[14] Angelo Filicetti, Michael Cody, and Scott Nielsen. Caribou conservation: Restoring trees on seismic lines in alberta, canada. *Forests*, 10:185, 02 2019.

[15] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels, 2020.

[16] Swarnendu Ghosh, Nibaran Das, Ishita Das, and Ujjwal Maulik. Understanding deep learning techniques for image segmentation. *CoRR*, abs/1907.06119, 2019.

[17] Geoffrey Hay and Guillermo Castilla. Geographic object-based image analysis (GEOBIA): A new name for a new discipline. *Object-Based Image Analysis - Spatial Concepts for Knowledge-driven Remote Sensing Applications*, pages 75–89, 01 2008.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[19] P V.C. Hough. Method and means for recognizing complex patterns, US Patent 3,069,654, 18 December 1962.

[20] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. Learning visual groups from co-occurrences in space and time. *CoRR*, abs/1511.06811, 2015.

[21] ISPRS. Isprs 2d semantic labeling contest. `http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html`. Accessed: 2020-07-29.

[22] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9865–9874, 2019.

[23] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[24] Thomas Joyce, Agisilaos Chartsias, and Sotirios Tsaftaris. Deep multi-class segmentation without ground-truth labels. In *Medical Imaging with Deep Learning*, July 2018.

[25] Asako Kanezaki. Unsupervised image segmentation by backpropagation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[27] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[28] Gustavo Lopes Queiroz, Gregory J. McDermid, Guillermo Castilla, Julia Linke, and Mir Mustafizur Rahman. Mapping coarse woody debris with random forest classification of centimetric aerial imagery. *Forests*, 10(6):471, May 2019.

[29] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016.

[30] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 11 2004.

[31] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[32] Takayasu Moriya, Holger R. Roth, Shota Nakamura, Hirohisa Oda, Kai Nagara, Masahiro Oda, and Kensaku Mori. Unsupervised segmentation of 3d medical images based on clustering and deep representation learning. *CoRR*, abs/1804.03830, 2018.

[33] Yassine Ouali, Céline Hudelot, and Myriam Tami. Autoregressive unsupervised image segmentation, 2020.

[34] Dimitrios Panagiotidis, Azadeh Abdollahnejad, Peter Surovy, and Karel Kuželka. Detection of fallen logs from high-resolution uav images. *New Zealand Journal of Forestry*, 49, 03 2019.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library, 2019.

[36] Judith M.S. Prewitt. Object enhancement and extraction. *Picture Processing and Psychopictorics*, pages 75–149, 1970.

[37] QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2009.

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *A talk at the Stanford Artificial Intelligence Project*, 1968.

[42] Cassidy K. van Rensen, Scott E. Nielsen, Barry White, Tim Vinge, and Victor J. Lieffers. Natural regeneration of forest vegetation on legacy seismic lines in boreal habitats in alberta's oil sands region. *Biological Conservation*, 184:127 – 135, 2015.

[43] Ali Varamesh and Tinne Tuytelaars. Mix'em: Unsupervised image classification using a mixture of embeddings. *arXiv preprint arXiv:2007.09502*, 2020.

[44] Leah A. Wasser. What is a CHM, DSM and DTM? About Gridded, Raster LiDAR Data. `https://www.neonscience.org/chm-dsm-dtm-gridded-lidar-data`. National Ecological Observatory Network. Accessed: 2020-09-21.

[45] Xide Xia and Brian Kulis. W-net: A deep model for fully unsupervised image segmentation. *CoRR*, abs/1711.08506, 2017.

[46] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. *CoRR*, abs/1604.03628, 2016.

[47] Evgenii Zheltonozhskii, Chaim Baskin, Alex M. Bronstein, and Avi Mendelson. Self-supervised learning for large-scale unsupervised image clustering, 2020.

List of Figures

List of Tables

A Further Examples

Image Clusters To give an intuition of the quality of SCAN as a clustering method, we illustrate some clusters that were grouped together by it in Figure 29. As one can see, the clusters are not perfectly separated but still represent a reasonable categorization of the images.

Segmentation with the Proposed Two-Step Procedure and SCAN In Figure 30, we provide additional examples for segmentations obtained with our two-step procedure. In particular, SCAN was used as a clustering method to assign semantic labels to the extracted segments.

Weakly Supervised Segmentation For the sake of completeness and for comparison, we also depict examples that were produced with a minimal amount of supervision in Figure 31. We trained a ResNet-18 with a two-layer classification head on 100 labeled samples. These samples contain image patches together with binary labels that indicate whether the patch depicts coarse woody debris or not.

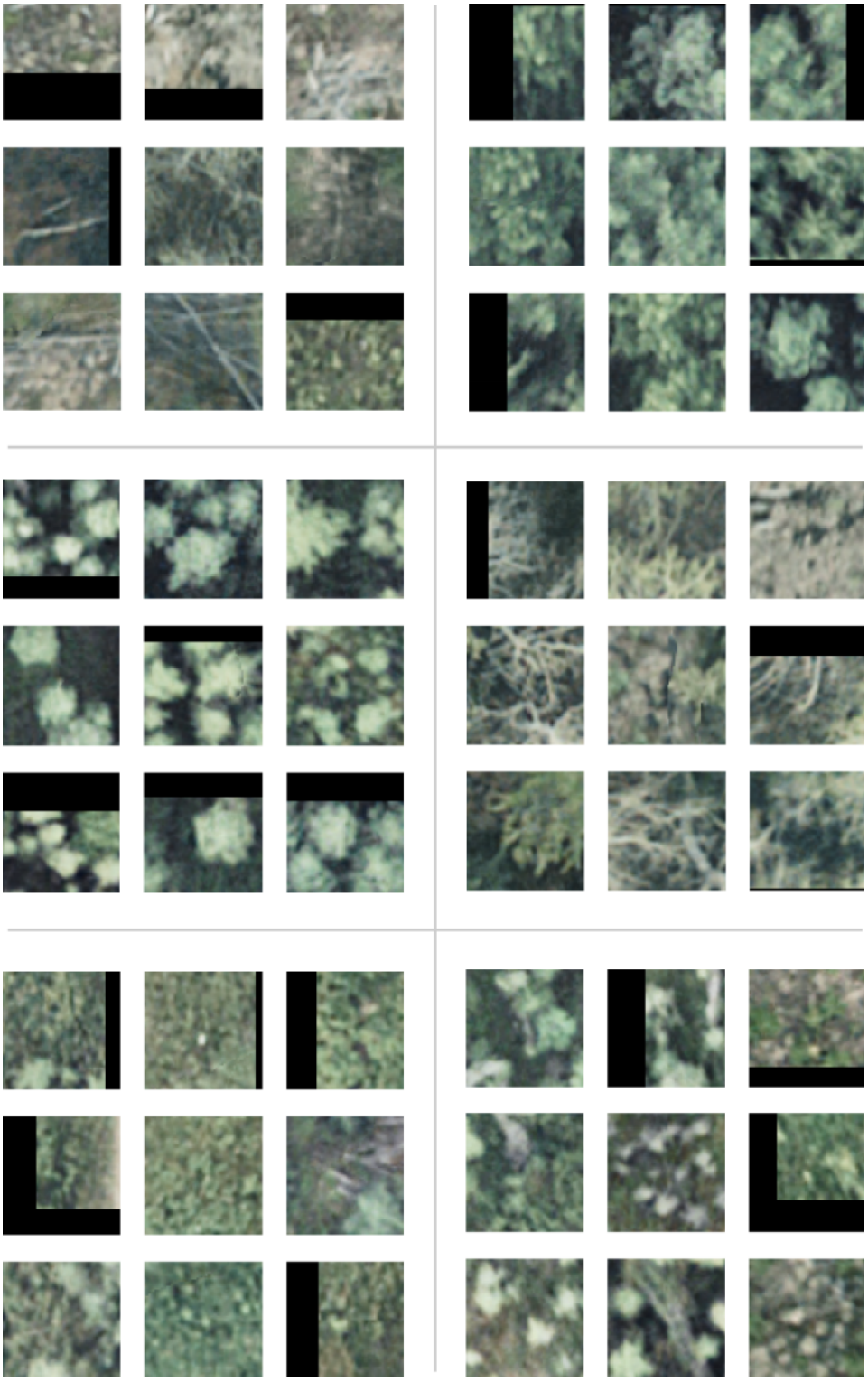


Figure 29: Examples for image clusters obtained with SCAN.

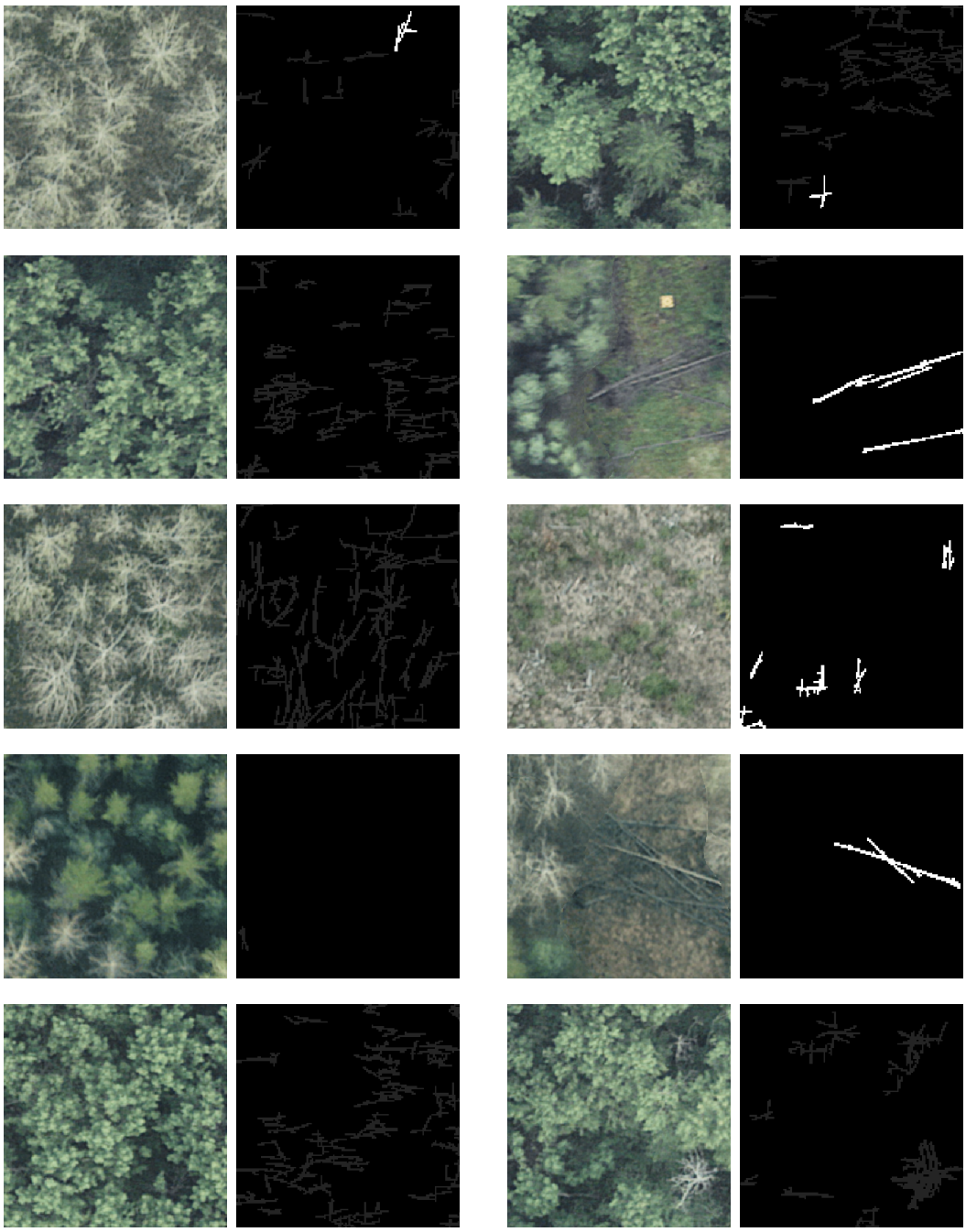


Figure 30: Randomly selected example images and their segmentations obtained with SCAN as clustering method in our two-step procedure for semantic segmentation. The white regions are the detected instances of coarse woody debris. The gray regions represent segments that were extracted as candidates by the segmentation algorithm, but eventually, not classified as coarse woody debris.

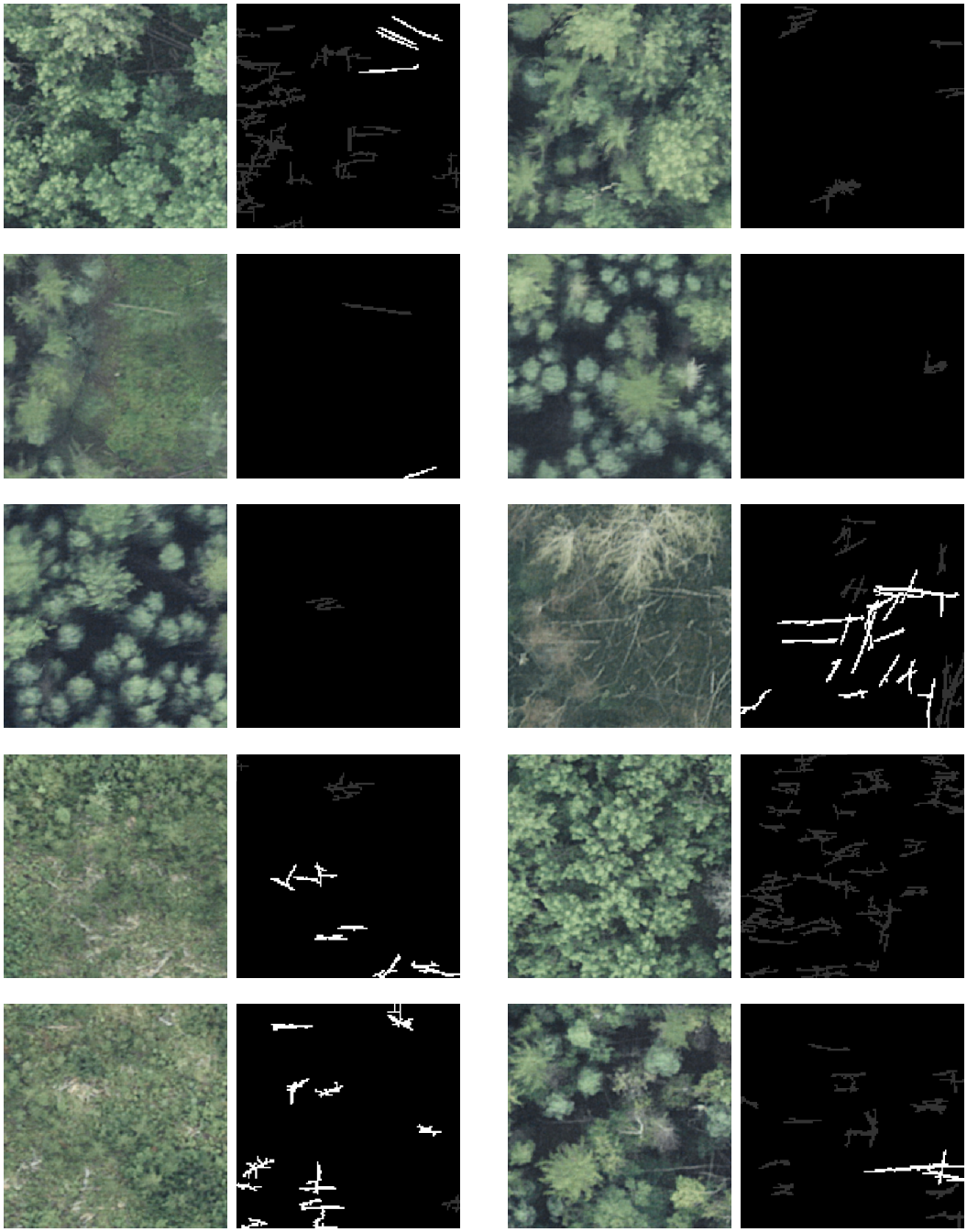


Figure 31: Randomly selected example images and their segmentations obtained with our two-step procedure and weak supervision. Once again, the white regions are the predicted instances of coarse woody debris. The gray regions represent candidate segments extracted by the segmentation algorithm that were not classified as coarse woody debris.

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.
This paper was not previously presented to another examination board and has not been published.

Munich, 28.10.2020

.....
Maximilian Bernhard