FEDERAL UNIVERSITY OF MINAS GERAIS

School of Engineering

Graduate Program in Electrical Engineering

João Pedro Klock Ferreira

General Unsupervised Semantic Segmentation Pipeline

Belo Horizonte

2023

João Pedro Klock Ferreira

General Unsupervised Semantic Segmentation Pipeline

Master dissertation submitted to the Examining Board designated by the Collegiate of the Graduate Program in Electrical Engineering of the Federal University of Minas Gerais, in fulfillment of the requirements for the degree of Master in Electrical Engineering.

Supervisor: Cristiano Leite de Castro

Belo Horizonte

2023

F383g

Ferreira, João Pedro Klock .
General unsupervised semantic segmentation pipeline [recurso eletrônico] / João Pedro Klock Ferreira. - 2023.
1 recurso online (113 f. : il., color.) : pdf.

Orientador: Cristiano Leite de Castro.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: 107 - 113.

Bibliografia: f. 95-106.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Redes neurais (Computação) - Teses. 3. Sensoriamento remoto - Teses. I. Castro, Cristiano Leite de. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela Bibliotecária Roseli Alves de Oliveira CRB/6 2121
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
ATA DA 1312ª DEFESA DE DISSERTAÇÃO DE MESTRADO
DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO do aluno **João Pedro Klock Ferreira** - registro de matrícula de número 2021679017. Às 11:00 horas do dia 12 do mês de maio de 2023, reuniu-se na Escola de Engenharia da UFMG a Comissão Examinadora da DISSERTAÇÃO DE MESTRADO para julgar, em exame final, o trabalho intitulado **"General Unsupervised Semantic Segmentation Pipeline"** da Área de Concentração em Sistemas de Computação e Telecomunicações, Linha de Pesquisa Inteligência Computacional. O Prof. Cristiano Leite de Castro, orientador do aluno, abriu a sessão apresentando os membros da Comissão e, dando continuidade aos trabalhos, informou aos presentes que, de acordo com o Regulamento do Programa no seu Art. 8.16, será considerado APROVADO na defesa da Dissertação de Mestrado o candidato que obtiver a aprovação unânime dos membros da Comissão Examinadora. Em seguida deu início à apresentação do trabalho pelo Candidato. Ao final da apresentação seguiu-se a arguição do candidato pelos examinadores. Logo após o término da arguição a Comissão Examinadora se reuniu, sem a presença do Candidato e do público, e elegeu o Prof. Cristiano Leite de Castro para presidir a fase de avaliação do trabalho, constituída de deliberação individual de APROVAÇÃO ou de REPROVAÇÃO e expedição do resultado final. As deliberações individuais de cada membro da Comissão Examinadora foram as seguintes:

Membro da Comissão Examinadora	Instituição de Origem	Deliberação
Prof. Dr. Cristiano Leite de Castro - Orientador	DEE (UFMG)	APROVADO
Prof. Dr. Elcio Hideiti Shiguemori	DCTA (Instituto de Estudos Avançados (IEAv))	APROVADO
Prof. Dr. Jefersson Alex dos Santos	Computer Science (University of Stirling)	APROVADO

Tendo como base as deliberações dos membros da Comissão Examinadora a Dissertação de Mestrado foi APROVADA. O resultado final foi comunicado publicamente ao Candidato pelo Presidente da Comissão, ressaltando que a obtenção do Grau de Mestre em ENGENHARIA ELÉTRICA fica condicionada à entrega do TEXTO FINAL da Dissertação de Mestrado. O Candidato terá um prazo máximo de 30 (trinta) dias, a partir desta data, para fazer as CORREÇÕES DE FORMA e entregar o texto final da Dissertação de Mestrado na secretaria do PPGEE/UFMG. As correções de forma exigidas pelos membros da Comissão Examinadora deverão ser registradas em um exemplar do texto da Dissertação de Mestrado, cuja verificação ficará sob a responsabilidade do Presidente da Banca Examinadora. Nada mais havendo a tratar o Presidente encerrou a reunião e lavrou a presente ATA, que será assinada pelos membros da Comissão Examinadora.

Belo Horizonte, 12 de maio de 2023.

ASSINATURA DA COMISSÃO EXAMINADORA

Documento assinado eletronicamente por **Cristiano Leite de Castro**, **Professor do Magistério Superior**, em 16/05/2023, às 17:15, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Jefersson Alex dos Santos**, **Membro**, em 17/05/2023, às 15:35, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Elcio Hideiti Shiguemori**, **Usuário Externo**, em 23/05/2023, às 14:28, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2285796** e o código CRC **03297BE2**.

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"GENERAL UNSUPERVISED SEMANTIC SEGMENTATION PIPELINE"

JOÃO PEDRO KLOCK FERREIRA

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica. Aprovada em 12 de maio de 2023. Por:

Prof. Dr. Cristiano Leite de Castro
DEE (UFMG) - Orientador

Prof. Dr. Elcio Hideiti Shiguemori
DCTA (Instituto de Estudos Avançados (IEAv))

Prof. Dr. Jefersson Alex dos Santos
Computer Science (University of Stirling)

Resumo

Neste trabalho apresentamos um pipeline de segmentação semântica não supervisionada baseado em Redes Neurais Convolucionais (CNNs), com foco em imagens de sensoriamento remoto. Nosso pipeline aprimora os artigos que representam o atual estado-da-arte na literatura, resultando em uma metodologia versátil que pode receber entradas supervisionadas, não supervisionadas e fracamente supervisionadas. Também propomos uma metodologia de geração automática de *scribbles* que é capaz de rotular semi-automaticamente grandes conjuntos de dados com supervisão mínima. Para acompanhar esta metodologia também propomos um classificador de *scribbles* e uma ferramenta de rotulagem de *scribbles*. E, finalmente, propomos duas aplicações do mundo real onde testamos as capacidades de nossa rede proposta. Os resultados em datasets de benchmarking mostram que nossa rede proposta pode ser competitiva com o estado da arte atual para métodos baseados em CNN, o gerador de scribble é capaz de fornecer *scribbles* significativos e relevantes para grandes conjuntos de dados, e nossas aplicações mostram uma variedade de possibilidades de uso para nossos rede.

Palavras-chave: segmentação semântica, não supervisionado, fracamente supervisionado, sensoriamento remoto, *scribbles*.

Abstract

In this work, we present a Convolutional Neural Network-based unsupervised semantic segmentation pipeline, with a focus on remote sensing imagery. Our pipeline improves upon the current state-of-art papers in literature, resulting in a versatile methodology that can take supervised, unsupervised, and weakly supervised inputs. We also propose an automatic scribble generation methodology that is able to semi-automatically label large datasets given minimal supervision. To accompany this methodology we also propose a scribble classifier and a scribble labeling tool. And finally, we propose two real-world applications where we test the capabilities of our proposed network. Results on benchmarking datasets show that our proposed network can be competitive with the current state of the art for CNN-based methods, the scribble generator is able to provide meaningful and relevant scribbles for large datasets, and our applications show a variety of possibilities of usage for our network.

Keywords: semantic segmentation, unsupervised, weakly supervised, remote sensing, scribbles.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Describing the contents of an image is as natural for humans as walking or breathing. Our developed brains and eyes, in a single look, can accurately detect and describe every aspect of an image in seconds. But for machines, this is a more complex problem. Starting with how machines understand images- sequences of zeros and ones in a grid- more complex and meaningful ways of looking into this data are required to make sense of what is happening in that rectangular box filled with colors. Digitally, images are described as a matrix of values, or pixels, where each value represents the color at that specific point. In the case of colored images, three values are necessary.

The main research field for images and how computers process them is Computer Vision, which consists of various techniques for processing digital images. Among such methods, there are those focused on describing the contents of images. Some examples are Image Classification, which consists of assigning simple labels indicating whether an image contains or not a given semantic element, such as animals, trees, objects, etc.; Object Detection, which solves a more complex problem of not only defining if an image contains a particular element, but also locating it within the image; and Semantic Segmentation, which consists of describing the contents of the entire image in a pixel level, indicating where each of the elements is and what they are, or, in other words, describing the semantics of the images.

In this work, a general pipeline for semantic segmentation is investigated. There are a variety of fields in which semantic segmentation can be applied, each with its sub-applications. To comprehend the importance of this technique, some of the main applications are listed here. In Medical Imaging, there are several usages, such as segmenting X-rays into different types of bones and tissues or microscope images of cells and tumors [Wang et al., 2022]. In Remote Sensing, applications include classifying crops, extracting buildings and roads, monitoring the environment, and others [Yuan et al., 2021]. As reported in Feng et al. [2021], semantic segmentation is one of the most crucial components of Autonomous Driving. In Biology and Agriculture, it can be used to differentiate species of plants [Milioto et al., 2018]. Its use for Face Recognition is extensively discussed in Meenpal et al. [2019]. It can categorize different items in Sales and similar applications [Rabinovich et al., 2007]. And along with these applications, there are many others since this is a versatile technique with endless possibilities.

Semantic segmentation can be understood as clustering parts of an image into distinct classes or categories according to their similarity [Thoma, 2016]. Before the advent of the Convolutional Neural Networks (CNNs), which compose the current state-of-art in this task, there were many different methods based on handcrafted features such as feature clustering, graph approaches, active contour models, and watershed algorithms, among many others [Thoma, 2016]. Most of these conventional methods follow a generative approach to learning the semantics contained in the image. Such an approach provides more information about the problem in question, allowing us to analyze, for example, the degree of class overlap.

After the first work that popularized the usage of CNNs [LeCun et al., 1998], the field of Deep Semantic Segmentation Networks (DSSNs) eventually arose [Shelhamer et al., 2017] and quickly reached the state-of-art in this task [Minaee et al., 2022]. The deep networks can learn the intrinsic features of the images with their filters, representing color and texture better than any other conventional feature extractor. Because of that, they reached new standards in every image processing task, e.g., image classification, object detection, and semantic segmentation. When using DSSNs, the previously primarily generative problem now becomes a discriminative approach, which consists of drawing boundaries in the data space by predicting labels for the data. Although generative models are usually more general and discriminative models are more application dependant, this specification allowed the networks to reach new standards for their domain of usage, generally surpassing and being preferred compared to the generative models [Ng and Jordan, 2002].

Therefore, one problem with these networks is the need for labeled ground truth data, that is, semantic masks describing the contents of thousands of images needed to teach them to perform an accurate segmentation. Even though they can complete this task precisely, they still require much training to learn how to segment. Supervised learning is teaching an algorithm a given task by providing a ground truth. By recalling the definition of semantic segmentation, to cluster parts of an image as belonging to the same class, to define the semantics of an image, or what is the content present in it, an expert must first know and name this content so that the algorithms may later label their segmented masks. But the main problem is that these ground truth labels are expensive to acquire since placing these labels is a complex and time-consuming task, sometimes even requiring money to be spent with experts.

The need for a lot of labeled data in supervised learning led researchers to search for other ways of teaching networks to perform semantic segmentation. Some learning approaches without needing a fully annotated ground truth were explored, such as using weak labels that describe only small pieces or fragments of images, such as scribbles in homogeneous areas and polygons or single points in some objects [Zhang et al., 2020]. This research field is called weak learning, using weak labels to teach algorithms instead of entire ground truths [Zhang et al., 2020]. But the main problem with this approach is that it also requires a vast amount of data to be labeled for the networks to learn correctly. Thus, for extensive datasets with several thousands of images, even with weak labels, the problem remains. The need for a lot of labeled data in supervised learning led researchers to search for other ways of teaching networks to perform semantic segmentation.

This drawback led researchers to explore more options, such as unsupervised learning: the ability to learn without labels. In this approach, the networks learn to perform the semantic segmentation by themselves, removing the need for data annotation [Thoma, 2016]. Another advantage of this approach, besides saving the time and labor of the annotation process, is that the networks also avoid human labeling errors, such as not depicting the best representation of the objects and regions. On the other hand, these networks many times are not able to completely represent the ground truth segmentation and usually have more error than the supervised approaches since the features learned may not indeed mean the semantic classes of the image as initially intended, which is why is a less used approach in general.

1.1 Main Contributions

Given the different types of learning described and motivated by the advantages of not needing a fully annotated dataset, this work proposes a new unsupervised semantic segmentation network that performs a clustering task, being able to learn standard features through an entire dataset, resulting in a powerful feature extractor for the proposed application. The clustering task is performed by grouping similar regions of images and teaching the network to classify these groups as the same class. By using a clustering approach, the network has the advantage of the generative models, that is, being able to describe better and represent the data, while also using having the benefit of the powerful training and feature extraction of the CNNs, discriminative models, taking the best of both worlds.

Our pipeline is composed of a feature extractor and two classifiers with q classes, where q is a large number, followed by an argmax operation to select the highest activated class for each pixel in each classifier. We then segment the images in superpixels and through a majority vote, attribute the most common class inside the superpixel for every pixel. We then use this as ground-truth and backpropagate to the network, using our novel cross-entropy loss weighted by each image batch. By using two classifiers and the same feature extractor, we are able to further generalize the features since their class is randomly assigned in each classifier.

Other works in literature explore unsupervised semantic segmentation with CNNs [Jing and Tian, 2021]. Still, unlike these works, the proposed network can also be extended to use annotated semantic masks to guide the learning process, applying to both the weak and fully supervised learning cases. This pipeline is defined as general. When using supervision, the main focus is not to learn to perform the most accurate semantic segmentation, since the semantic mask is only used to group pixels and the classes are discovered by the network itself, but to create a more robust representation of the features, allowing the network to expand supervised classes into more than one cluster, being able to even take into account their overlap and avoiding annotation noise. This versatility to alternate between supervised and unsupervised cases, along with the robust feature representations, are among the main contributions of this work.

But since in both weak and fully supervised cases, there is a need for a lot of annotated data, a novel methodology for labeling datasets with weak labels is also proposed, which drastically reduces the annotation effort, even in the already low-effort weakly-supervised case, and justifies the use of the proposed network with human supervision. In this pipeline, the proposition is that by grouping similar regions in images, the same premise as in the clustering task, it is possible to automatically draw scribbles in well-defined and homogeneous areas of an entire dataset of images, label a small set of these automatically generated scribbles, and train a classifier to label the remaining, resulting in a semi-automatic labeling process. Accordingly, the main advantage of the proposed unsupervised semantic segmentation methodology, which is to dispose of the effort of labeling a dataset, can still be reached in a supervised context by reducing this effort to the bare minimum. The process is also facilitated by our provided scribble labeling tool, which allows the user to easily draw, edit, remove and pick a class for scribbles.

Finally, given the clustering task that this work proposes to solve, the chosen application for the work to be developed and tested are remote sensing tasks, which usually have well-defined regions and objects throughout the image and can be used to solve a variety of different problems, therefore composing a challenging domain for this work. In our experiments, along with the semantic segmentation task, we propose two usages of our network in remote sensing applications, investigating the complexity and usability of the learned features. Given these considerations, our main contributions can be summarized:

- An unsupervised deep semantic segmentation network that can learn complex semantic patterns;
- An analysis methodology for two examples of remote sensing applications where this

network can be applied;

- An automatic scribble generator;
- A semi-supervised labeling pipeline that requires only a small set of images to be labeled;
- A graphical user interface to visualize, draw, edit, remove, and annotate scribbles manually;
- A published paper with part of our results: Ferreira et al. [2022]

The remaining of this document is organized as follows: in Chapter 2, a state-of-art review for both deep unsupervised semantic segmentation and weak label generation are presented and compared, along with the primary motivations for this work; Chapter 3 introduces the proposed methodologies for both the unsupervised semantic segmentation pipeline and the semi-automatic weak labeling process. In Chapter 4, the results for the pipelines and the applications are presented and discussed; Finally, Chapter 5 offers the conclusion for this work.

Chapter 2

Related Works

This chapter introduces the Unsupervised Semantic Segmentation algorithms, from the first algorithms developed to the latest state-of-art in literature. A review of the Weak Label Generation methodologies is also presented, ending with the main contributions of this work to current and future literature.

2.1 Unsupervised Semantic Segmentation

Before explaining the unsupervised networks, it is first essential to understand the history of the supervised semantic segmentation networks since their building blocks and architectures have shaped everything used until today in semantic segmentation. This section starts with the history of the most common supervised methods, followed by the current state-of-art of the unsupervised methods.

2.1.1 History of supervised segmentation networks

The first work to propose a CNN was an algorithm based on the hierarchical receptive field model of the visual cortex, called "Neocognitron," dated from 1980 [Fukushima, 1980]. But CNNs would not become popular until 1998, when the LeNet architecture was proposed, applied to digit recognition [LeCun et al., 1998], which is a composition of convolutional layers, where a filter would convolve with a standard 2D neural network layer; non-linear layers, which apply an activation function on the resulting feature map of another layer; and pooling layers, that reduce the dimension of 2D layers based in local statistical information. Compressing the images in resulting features is also called an encoder process.

Although these networks had promising results, the computational effort required to use these networks had still not been achieved by then. But with the advent of Deep Learning techniques and the usage of Graphic Processing Units (GPUs) to train models, in 2012, the AlexNet network was proposed [Krizhevsky et al., 2012], which was a wider and deeper version of the LeNet, being able to learn and recognize more complex features and objects.

Afterward, a diversity of famous CNNs architectures were proposed. Some examples are: VGGNet [Simonyan Karen and Zisserman Andrew, 2015], or VGG-16, which is also very similar to the AlexNet architecture, but it is more extensive, with 16 layers, and uses only 3×3 filters to perform the convolution operations; ResNet [He et al., 2016] which introduced the skip connections, a way to connect the first layers with the last ones and avoid the vanishing gradient problem, where deep layers had more challenging time to learn; GoogleLeNet [Fairclough et al., 2015] which introduced two contributtions, the inception module: a way to concatenating different convolutional filters and reducing the number of features and operations, and the usage of batch normalization during training, the normalization of the features between layers along the training; MobiletNet [Howard et al., 2017], which used depthwise convolutions, a new type of convolution that uses much less parameters, drastically reducing the size of large networks while still achieving the same results, and allowing deep networks to be suited for mobile applications; and finally DenseNet [Huang et al., 2016], which is a more profound and broader network that connects every layer to each other, using less parameters and making a better use of the features, thus avoiding vanishing gradients.

Each of these networks had its advantages and changed the history of CNNs. But as far as semantic segmentation is concerned, the most significant change occurred when the Fully Convolutional Network (FCN) architecture arrived [Shelhamer et al., 2017], one of the first DSSNs, a deep learning network specifically for semantic segmentation. In this network, the authors use the VGGNet architecture as an encoder, and at the end of the network, the output features are upsampled to the input image dimension by using a bilinear interpolation, allowing the usage of a classifier that acts directly in each pixel of the image. Along with the upsampling step, the authors used skip connections between lower and higher layers, similar to ResNet. This allowed the network combines both the more complex semantic information from the final layers and the less complicated appearance features from the first layers, producing a more accurate segmentation.

Among many supervised DSSNs that arise later, some specific networks deserve attention. The first one is U-Net [Ronneberger et al., 2015], a network that came before FCN and used two VGGNets concatenated in an U shape. The idea is similar to FCN, but the second network performs multiple upsamples instead of one. Another difference is that instead of bilinear interpolation, the upsamples are performed using deconvolutions - filters with learnable weights that can increase the dimension of the output, making the upsampling step more robust. This type of architecture is called encoder-decoder architecture. They also extend the concept of skip connections; instead of the traditional connections between deeper and shallower layers, the authors concatenate the feature layers from the encoder network to their symmetrical, or equivalent in size, layers in the decoder network during the upsample convolutions, avoiding losing patterns in the deconvolution and turning the upsampling process becomes even more robust and less prone to vanishing gradients.

Another widely used network is SegNet [Badrinarayanan et al., 2017]. This network is similar to U-Net: both have convolutional and deconvolutional networks concatenated, but SegNet introduces a connection between the pooling layers from the convolutional step into the deconvolutional step, which maps the pooling indexes and uses them to perform an upsample, in a process called unpooling. This process eliminates the need for skip connections, thus eliminating the need to learn weights in the upsampling process and being more robust than bilinear interpolation. The result is a smaller network with fewer parameters, as it is wholly made of convolutional layers, achieving results equally good or even better than FCN and U-Net.

Another type of network that greatly impacted semantic segmentation is the Dilated Convolutional Model, with the principal works being the DeepLab models [Chen et al., 2018]. These networks introduced the dilated convolutional layer, which is the same as a convolution but adds a space between the kernel weights. This helps during the step of decreasing the resolution of the images in the encoder step. The networks also introduced the Atrous Spatial Pyramid Pooling (ASPP), which uses features from multiple resolutions at specific feature layers, helping to capture objects and image context at various scales. And it also introduced the usage of fully Conditional Random Fields (CRFs) to improve the localization of object boundaries.

Along with this described network, which is the most used, other types of networks include Multi-Scale and Pyramid Network-based models [Lin et al., 2017], Attention base models [Chen et al., 2016], Generative and Adversarial models [Luc et al., 2016, Souly et al., 2017], Recurrent Neural Network models [He et al., 2017], among several others. This comprises most of the history of the supervised DSSNs, which originated the current state-of-art models in fully and weakly supervised learning.

2.1.2 State-of-art for unsupervised segmentation networks

Regarding unsupervised learning, the models differ slightly from the supervised ones since they are built to learn how to segment the images independently. This literature review will focus on works that: 1 - use CNNs; 2 - uses only images with the primary Red, Green, and Blue (RGB) colors; and 3 - were relevant to the current state-of-art for this more specific problem.

One of the first works that greatly impacted unsupervised segmentation was called Deep Cluster [Caron et al., 2018]. In this work, the authors took advantage of features in the final layer of the model. After the end of the decoder step, they apply a Principal Component Analysis (PCA) technique to reduce the dimensionality and then cluster the features using the K-Means algorithm, with a few additional constraints to avoid trivial solutions. The clusters generated are used as pseudo-labels for each pixel and backpropagated with a standard classification loss for the network. This network was initially proposed for image classification. Still, the authors extended it for the semantic segmentation problem by using the previously trained classifiers to classify pixels individually. At the time, it achieved good results, also being a generic way of solving unsupervised CNN training.

Another work that set new baselines in unsupervised semantic segmentation was the Invariant Information Clustering (IIC) [Ji et al., 2019]. In this work, the idea is to use pair of images: a random image from the training set and the same image but with some perturbation (i.e., data augmentation). Their network comprises a CNN to extract features and a fully connected layer to predict the probabilities. Then, knowing that both images have the same clusters, that is, their probability distributions for each class should be close, they use an equation that measures the mutual information of the distributions and models it as a loss function. The idea is to maximize the mutual information between given pairs to assign similar images to the same class. They apply this pipeline in different image patches to perform semantic segmentation with minor modifications. Finally, they also propose using a second classifier with more classes than in the actual dataset, called distractor classes. In this second classifier, the proposition is to use an over-clustering (more classes than ground truth) approach, explaining that the desired relevant classes may contain errors and overlaps and that by adding this second classifier, they can still benefit from the context in these distractors classes, so that the network will still learn to distinguish features while paying attention to the desired segmentation, even with noisy classes.

Another relevant work was the W-Net [Xia and Kulis, 2017]. Based on the U-Net, this network works by coupling two U-Net architectures, taking the shape of a W and thus originating its name. The first U-Net is an encoder that will perform the semantic segmentation of the image, and the second U-Net tries to reconstruct the original image by using the output of the encoder. Two losses are optimized, the mean square error between the original and reconstructed image and a loss based on the normalized cut algorithm [Shi and Malik, 2000], which is a graph-based clustering algorithm. To understand this second loss, the authors initialized a generic number of classes K, and at the output of the first U-Net, the argmax of the activations is taken to predict each of the K classes activated the most for each pixel, thus becoming the class of the pixel; this loss is a soft version of the Normalized Cut algorithm, which takes into account the argmax function, is differentiable and able to backpropagate. A post-processing step is also performed, where the segments are smoothed using Fully Connected Random Fields. Then, a hierarchical segmentation method groups and combines smaller segments, generating the final semantic segmentation with a specific class for each pixel. This class is then used as ground truth and backpropagated to the network during training.

Besides these works based on classic CNNs, recently, approaches based on visual transformers also arose, which are different from the three initial categories, but currently compose the state-of-art in unsupervised semantic segmentation. This first algorithm works in two steps [Hamilton et al., 2022]. First, they perform a self-supervised feature learning where patches from the same image and different images go through the same backbone to train a segmentation head, and the loss function maximizes the similarity of features from the same image while minimizing the similarity between features from different images. In the end, similar regions of similar images will have closer feature vectors, and this can be used to verify if two images are from a similar class. Then a lookup table is constructed for the entire dataset with every class. The batches for training are composed of random images, and for each image, its 7 K-Nearest Neighbors (KNN) using the lookup table. During training, there are three losses, one that evaluates the correspondence of the features within the image, one to evaluates the correspondence with the features of one of the k-neighbors, and one to evaluates the correspondence with the features of a random image from the batch. The first two provide a positive signal, while the last provides a negative signal. In the end, the network will have learned the semantic segmentation of the dataset.

These four works compose the base of unsupervised semantic segmentation. Many works that arose later than the first three methodologies were created as variations and recent works are being created as variations of the fourth. And thus, the three main methodologies are here classified as four Unsupervised Methodologies (UM):

- $UM1$ - Methods based in Information Theory [Ji et al., 2019]
- $UM2$ - Methods based in structured clustering [Caron et al., 2018]
- $UM3$ - Methods based in spatial clustering post-process (after argmax) [Xia and Kulis, 2017]
- $UM4$ - Methods based in visual transformers [Hamilton et al., 2022]

Recently, other methodologies arose that are worth mentioning, but despite being different in some aspects, most of them can be categorized within these four initial works. Some of the works include:

1. one work that segments an image into superpixels, followed by the extraction of low-level (statistical) features that are fed to a Stacked Autoencoder (SAE), whose

purpose is to generate high-level features, and finally, the high-level features are clustered in a post-processing step using a graph-based superpixel clustering algorithm [Jiao et al., 2020] - similar to $UM3$;

2. another work, in a process similar to the IIC algorithm [Ji et al., 2019], uses as input two images, one which has photometric and geometric transformations as data augmentation, and feeds these images to the same network, using a modified loss function that tries to minimize the invariance to photometric transformations and maximize the equivariance to geometric transformations between images [Cho et al., 2021] - this can be categorized in $UM1$;

3. a work that extends the IIC algorithm [Ji et al., 2019] to perform a pixel-wise evaluation instead of using patches by extracting local features (from the first layers of the network) and using them in the mutual information maximization step to make the predictions closer to these local features instead of closer to each other [Harb and Knöbelreiter, 2021] - this is categorized in $UM1$.

Returning to the three unsupervised segmentation methodologies, the pipeline proposed in our study is based on the $UM3$. This idea of using a high number of classes and performing a refinement after taking the argmax value of the class activations proved to be strong, and many works after this improved this idea. One of the critical works that improved upon this idea proposed a more simple way of performing the same segmentation as $UM3$. Using a simple and generic CNN to extract features, the author suggests that q classes are initialized and that the argmax of these classes is taken to classify the pixels. Then, instead of using the Normalized Cut algorithm during training, a simple superpixel segmentation with the Slic algorithm is used to enforce that the pixels within a superpixel belong to the same class. Then, backpropagate a simple softmax classification loss to the network, comparing the obtained and superpixel-refined segmentations. By pre-storing the superpixel segmentation of the images in memory, this process becomes more straightforward and faster than the W-Net, since the network architecture and clustering steps after argmax are more straightforward while still achieving good segmentation results [Kanezaki, 2018].

The simplicity of this methodology allowed for more robust versions of this pipeline to take place. One of the first proposed the Mumford-Shah Loss [Kim and Ye, 2020], a loss based on the Mumford-Shah functional that minimizes the pixel variance inside the segments, acting as a regularizer during training. This loss can be directly coupled with other losses, improving existing segmentation algorithms, such as the segmentation from Kanezaki [2018].

The authors proposed two modifications to the original work in another work [Saha et al., 2019]. In contrast to the simple feature extractor from the original work of Kanezaki

[2018], the authors proposed the usage of a more robust feature extractor, the VGGNet layers, pre-trained in the ImageNet dataset, with the weights frozen, along with a few of the features blocks from the original work, making only these feature blocks trainable. And another modification is that instead of the superpixel refinement, they propose a segment score-based refinement to avoid over-segmented clusters. This is performed by checking the 8-neighbors for every pixel to determine connected segments. Then, if two segments have a different class and the area of one is more extensive than 50% of the size of the other, the smaller segment is merged with, the larger one. Both modifications improve the quality of the final obtained segmentation related to the original work.

The authors from another paper proposed a similar pipeline [Barthakur and Sarma, 2019], with a few modifications of exchanging the feature extractor to the SegNet architecture, and instead of superpixels, they convert the image to the Lab color space and cluster the image, then using the obtained segments as refinement class to the network. This work is generally closer to the $UM1$ category, although based on a $UM3$ method.

In a crop monitoring application work [Bhatt et al., 2019], the authors use the FCN architecture to extract features. After obtaining the labels through the argmax step, they proposed using a CRF post-processing step, which increases the segmentation accuracy by refining the boundaries. The clustering algorithm used is the Normalized-cut, and after the CRF, the remaining steps of the algorithm are the same - reassigning the labels and backpropagating the error. This work is closer to the W-Net than Kanezaki [2018].

One of the works tried to incorporate the idea of multi-temporal segmentation [Saha et al., 2020]. Besides the previous steps of the methodology, they prose the usage of a new loss that detects object segments from individual images and establishes a correspondence between distinct multi-temporal images. To perform that, the authors process images t and $t + 1$ at the same time, at any time t, and calculate the loss for both images using the same segmentation, guaranteeing that both will have the same clusters since images close in time most likely does not change much. Overall, the process is the same as in the original work of Kanezaki [2018], but with the addition of processing two images simultaneously.

In another work called SEEK [Ilyas et al., 2020], the authors exchanged the feature extractor by the SE-Net [Hu et al., 2020], a squeeze and excite architecture, which performs a feature calibration. They also traded the superpixel segmentation from the original Slic algorithm to the Felzenswalb algorithm [Felzenszwalb and Huttenlocher, 2004], a graph-based algorithm. Along with these changes, they also incorporated a K-means clustering step after the argmax to remove smaller segments and reduce the over-segmentation.

One author proposed to exchange the feature extractor for an encoder-decoder variation, where the basic blocks include residual connections, as in a ResNet [Khan and

Yang, 2020]. In this methodology, after training the network in the same way as Kanezaki [2018], except for the different feature extractors, the authors perform a post-processing step, extracting different color, texture, and spatial features from each superpixel. In other words, they merge superpixels with similar characteristics, creating more consistent areas for the final segmentation. These features are used in a Region Adjacency Graph, which is used in a Region Merging Algorithm.

Another author also proposed to change the feature extractor to an encoder-decoder architecture [Zhou and Wei, 2020]. Instead of residual blocks, they used the original blocks of Kanezaki [2018], composed of convolutional, ReLU, and batch normalization layers. They also added a novel block at the end of the pipeline, a Deep Subclustering Network (DCS) block, which clusters the features from the previous layers into several subclusters and then feeds the clusterized image to a convolutional layer. This layer works as a preprocessing step, pre-clustering the image in the feature space and improving the argmax semantic segmentation step, which will have less influence over the final result.

Finally, one work used an autoencoder architecture to solve the same problem, where they reconstructed the image at the end of the pipeline and applied the superpixel refinement in the middle, after the encoder, and before the decoder Lin et al. [2020]. The main difference in this approach is that they reconstructed a smoothed version of the image and used it to extract the superpixels. They also use two other losses, the original cross-entropy loss, and a proposed Superpixel Similarity Loss, in which they create a similarity matrix between close superpixels based on the idea that closer superpixels should have similar features, then try to minimize the sum of the similarities.

Given all the contributions above, the current state-of-art provides good semantic segmentation results considering an unsupervised problem. Most of these techniques can already achieve rich semantic mappings regarding the classes. Many times, considering the actual classes to be labeled by humans, they are subject to error, which can sometimes be reflected in the model. One of the most significant possibilities of these networks that learn the classes and structures by themselves is the ability to find hidden and subjective meanings in the data that a human could potentially miss. Therefore, the current problem in literature is not the supervised results metrics but the lack of applications that use these semantic structures in elaborate ways. There is much potential for the capabilities of Unsupervised DSSNs that are currently not well explored, which this work intends to study.

2.2 Weak Label Generation

As of the writing of this work, the most accurate and popular semantic segmentation method is DSSNs [Minaee et al., 2022]. In this context, just as many other modern machine learning systems that can learn highly complex tasks, they also require a large quantity of data to generalize the model for the most significant number of cases. Fortunately, the modern world has a large amount of data available. The problem is there is a high cost for labeling this data, specifically for semantic segmentation tasks, which depend on the availability of a fully annotated dataset with segmented masks in which every pixel in the image has its label. Making these masks can be both tedious and time-consuming, sometimes even requiring the help of experts to provide the correct annotations, therefore also becoming expensive. Furthermore, in remote sensing, this problem extends to the amount of different and unpredictable content in images, which contain varying information and a wide range of possible semantics.

Regarding data labeling in general, the three main problems were summarized by Roh et al. [2021]. The first one is the insufficient quantity of labeled data; when machine learning techniques are initially used in new applications or industries, there is often insufficient training data available to apply traditional machine learning processes. The second problem is insufficient subject-matter expertise to label data; when labeling training data requires relevant specific knowledge, creating a usable training dataset can quickly become prohibitively expensive. And the third problem is insufficient time to label and prepare data: Most of the time required to implement machine learning is spent preparing datasets. When an industry or research field deals with problems that are, by nature, rapidly evolving, it can be impossible to collect and organize data quickly enough for results to be beneficial in real-world applications.

The labeling problem mainly affects the classical type of learning: supervised learning, in which given input data and the desired output, the model is trained to map the input to the output. Aside from that, there are also several other types of learning, some helpful to dealing with fewer labels [Sarker, 2021, Sah, 2020], such as the opposite of supervised learning is the other classical method, unsupervised learning, in which no output is provided, and the model learns from the data. But in most applications, supervised methodologies perform better than unsupervised ones since the training to obtain the desired output can be considerably accelerated by giving the correct labels.

Due to the problems above, many researchers have tried to find ways of generating ground truth data more efficiently. Some learning types in the literature deal with this problem differently. The first is semi-supervised learning, which lies between supervised and unsupervised domains. In this type of learning, a mix of labeled and unlabeled data is used, where a model is trained using labeled data, and this model is used to classify the unlabeled data. Afterward, both are used to train the model with the whole dataset. Another approach can be to obtain a feature representation for the unlabeled data, which can be made by using an auto-encoder, and use it to replace or enhance the labeled data. And another type is self-supervised learning, which uses naturally existing supervision signals obtained from the data structure as labels to train the model, such as rotations of the image or horizontal/vertical flips.

The problem with these types of learning is that neither deals with the labeling problem for a vast amount of data, except maybe the self-taught learning. However, since the labels are placed automatically, it is still inaccurate. Finally, one more type of learning is weakly-supervised learning [Ratner et al., 2017a, Zhou, 2018]. The main idea is to obtain a more straightforward and cheaper way to annotate datasets through weak labels, that is, labeling a dataset with little cost, which can guide the model training like supervised learning. For disclosure, there are other types of learning, such as transfer, active, zero/one/few-shot, and reinforcement learning, which will not be detailed here.

To better detail, weak supervision is a form of coarse or inaccurate supervision. It is about leveraging higher-level or noisier input (simple form of label data) from Subject Matter Experts (SMEs) to train the model [Ratner et al., 2017a]. The primary motivation behind weak supervision is to mitigate the need for labeled training data. Unlike traditional, semi-supervised and transfer learning approaches, weak supervision uses lower quality labels more efficiently or at a higher abstraction level, being able to get cheaper labels from non-experts, to obtain higher-level supervision over unlabeled data from SMEs, and even to use one or more pre-trained models to provide supervision.

In a formal definition, consider a set of unlabeled data $X_u = x_1, \ldots, x_N$, and one or more weak supervision sources $\tilde{p}_i(y|x), i = 1M$, provided by an SME, such that each one is comprised of:

- A coverage set C_i, which is a set of points x over which it is defined
- An accuracy, which is defined as the expected probability of the label y^* over the coverage set

These weak supervision sources are called weak labels. For instance, consider the case of scribbles, where an example of an image with a complete and expensive per-pixel annotation is given, along with a cheap, weakly labeled scribble annotation. Each scribble drawn over an image has its associated pixels, which would be its coverage set and a class label attained to that particular scribble, which would be its expected probability.

According to Ratner et al. [2017a], weak label distributions serve as a way for human supervision to be provided more cheaply and efficiently, either by providing:

- Higher-level, less precise supervision (e.g., heuristic rules, expected label distributions)
- Cheaper, lower-quality supervision (e.g., crowdsourcing)

- Taking opportunistic advantage of existing resources (e.g., knowledge bases, pre-trained models)

With that in mind, three main types of weak labels are now defined, based on previous knowledge in literature [Ratner et al., 2017a, VERMA, 2021, Wikipedia, 2021].

The first one is Imprecise or Inexact labels. This type of label can be obtained by an active learning approach where the subject matter expertise annotates data with less precise labels. There are many examples of this type of label, such as physics-based constraints [Stewart and Ermon, 2017], output constraints on the execution of logical forms [Clarke et al., 2010, Guu et al., 2017], object detectors [Tighe and Lazebnik, 2013, Hariharan et al., 2014, Dai et al., 2016], object bounding boxes [Zhu et al., 2014, Chang et al., 2014, Hua et al., 2022], image-level class labels [Pathak et al., 2015, Pourian et al., 2015, Shi et al., 2017, Shimoda and Yanai, 2016], annotated superpixels [Ma et al., 2019], scribbles [Lin et al., 2016, Tang et al., 2018a,b, Hua et al., 2022], and even single pixels [Wang et al., 2020b, Hua et al., 2022].

The second one is Inaccurate labels. This type of label can be obtained by semi-supervised learning where the labels on the data sets can be of lower quality, brought by some expensive means like crowdsourcing. Such labels are numerous but not perfectly accurate. Some examples are: crowdsourcing labels [Ruvolo et al., 2015, Berend and Kontorovich, 2014, Zhang et al., 2014], heuristic rules outputs [Alfonseca et al., 2012, Ru et al., 2018, Roth and Klakow, 2013, Laine and Aila, 2017, Jin et al., 2014, 2017], noisy measurements [Bootkrajang and Kabán, 2012, Liang et al., 2009, Mann and McCallum, 2010], weak classifiers [Ratner et al., 2016], user-provided labels, such as binary labels indicating the presence of elements in images [Liang et al., 2009, Mann and McCallum, 2010, Ratner et al., 2017b, Mithun et al., 2019, Wang et al., 2020b] or feature expectations Liang et al. [2009], Mann and McCallum [2010], distributions or measurements [Druck et al., 2009].

And the third one is Existing labels. This type of label can be obtained from existing resources like knowledge bases, alternative data for training, or the data used in pre-trained models. A few examples are: distant supervision approaches [Mintz et al., 2009, Alfonseca et al., 2012, Ru et al., 2018, Roth and Klakow, 2013] and data transformations (such as data augmentation).

Although the annotation via weak supervision is less time-consuming, in most cases, the interference of an expert who must go through all the images of the dataset to insert the weak labels is still necessary, which is often a problem for large datasets containing tens/hundreds of thousands of images. This has been dealt with before in semi-supervised image classification, with the strategy of labeling a minor part of the dataset, training a model with fewer images, and then classifying the remaining data [Zhou, 2018, Wang et al., 2020a, Castillo-Navarro et al., 2021]. However, this cannot be used when using weak labels for semantic segmentation because the labels are more complex than just assigning a class for an image.

To address this problem, automatic weak label generation has been used before in image colorization, where automatic scribbles would be generated and manually classified to colorize images [Ding et al., 2012]. But the labels would still have to be manually allocated, meaning going through the entire dataset and classifying each scribble individually, therefore not solving the problem of large datasets. In a similar work, the authors propose drawing regions for scribbles [Batra et al., 2011], but the same issue of manual labeling remains.

While many works of semantic segmentation use weak labels to address large datasets, there is still a gap in the literature for works that improve the labeling process. Since one is designed to help with the other, authors many times do not consider the problem of extensive datasets. Some works tried different techniques that could be used in this context, but none has proposed anything to specifically mitigate this problem, which this work proposes to address.

2.3 Our Work

Given the current state of the art for unsupervised semantic segmentation and automatic weak label generation, this work brings a few contributions. First, this work proposes a new unsupervised segmentation pipeline whose focus is not to obtain the best results for supervised problems but to extract and learn the most relevant image semantics to be applied to remote sensing problems. To do this, we propose a network based on the work of Kanezaki [2018], which fits in the $UM3$ category (based in spatial clustering post-process), and is simple, allowing for different customizations, as proved by various works derived from this specific one. We generically built our pipeline so that it can perform semantic segmentation in an unsupervised manner and adopt complete and weak levels of supervision, therefore being a generalist pipeline.

As for the weak labeling problem, this work presents a solution to generate semantic segmentation masks using only a small set of a large dataset. Based on ideas from both Ding et al. [2012] and Batra et al. [2011], by combining weak and semi-supervised learning concepts, we propose a pipeline to automatically generate scribbles in images based on regions with similar content. Then, the expert must annotate only a small set of the scribbles, i.e., selecting the scribble label, which we also provide a tool to perform. A classifier will learn from these human-labeled scribbles and automatically classify (label) the remaining scribbles.

Finally, the main focus of this work is remote sensing applications, which by themselves pose many problems, such as a high level of context in the images, different levels of spatial resolution depending on the satellite from where the images were acquired, the usually large size of the images, along with other problems. And in our case, we chose to use only the RGB bands so that the pipeline could be applied in other areas.

Chapter 3

Methodology and Expected Results

This work proposes a general unsupervised semantic segmentation pipeline and a semi-automatic scribble generation pipeline. In this chapter, each methodology will be explained and discussed regarding architectural details. As for the proposed applications, these will be detailed in the results section.

For the unsupervised semantic segmentation pipeline, our hypothesis is to prove that, given the current literature, we can propose advancements to create a network capable of extracting meaningful semantics from remote sensing images without any source of supervision. Therefore, this work will discuss every step of the proposed pipeline in depth, explaining how each step may impact and contribute to the desired results.

The scribble generator's primary goal is to provide a valuable and low-cost way of labeling datasets, even lower than traditional weak supervision. From a practical perspective, this would allow quick annotations of extensive datasets, resulting in fewer time expenses. Our hypothesis is to prove that: a. through handcrafted features extracted from images, we can generate scribbles in homogeneous regions of such images; and b. given a dataset of images with auto-generated scribbles, we can correlate features from multiple images, allowing a classifier to learn these representations and classify scribbles. With this last hypothesis proven, given a small annotated set of images with scribbles, we could automatically classify an extensive collection of images without manual annotation.

Finally, this work intends to validate our solution in remote sensing imagery applications, that is, aerial and satellite images, which present their specific problems, such as a high amount of semantic information and possible image distortions. For that, we propose a set of applications that will be better explained and discussed in Chapter 5.

3.1 Unsupervised Semantic Segmentation Pipeline

We start with the standard pipeline for every semantic segmentation methodology to understand the unsupervised pipeline. They are composed of 3 blocks: the first for preprocessing the input image, which includes normalizations, resizes, and any data augmentation; the second to extract features from the image, which is usually any CNN feature extractor, such as the decoder step of any network, e.g., LeNet, VGGNet, ResNet, etc.; and the third to classify the pixels, which is usually a softmax classifier. This can be illustrated in Figure 1.

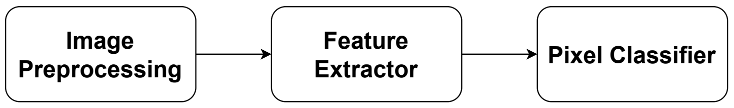


Figure 1 – Common pipeline in semantic segmentation solutions, composed of 1 - an input image preprocessing, 2 - a feature extractor, and 3 - a pixel classifier.

Source: produced by the author.

To make a fully unsupervised pipeline, similar to W-Net [Xia and Kulis, 2017], the work of Kanezaki [2018] proposed that, at the end of the semantic segmentation pipeline, a superpixel refinement step takes place, in which the image is segmented into superpixels, simulating a ground truth where the pixels inside each superpixel are considered to be in the same class. To ensure that, the authors initialize the network with a high number of classes q, and during the inference step, each pixel will be given a random class. Still, for each superpixel, the most frequent class inside that superpixel is chosen and spread through every pixel. This means that after propagating through the feature extractor, every superpixel region will be attributed to one of the random q classes. Since the number of classes is unknown, q must be high enough to comprise all possible semantics in the dataset. After this superpixel refinement, the network considers this the optimal output and backpropagates the error to the rest of the network as a common supervised loss. Usually, at the end of the training, only a few optimal classes learned by the network itself will be active, making the whole process an unsupervised semantic segmentation. This process is illustrated in Figure 2.

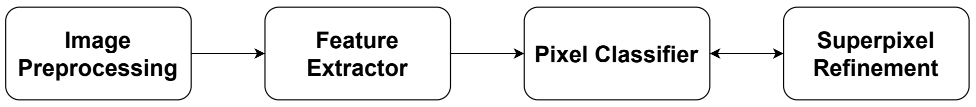


Figure 2 – Unsupervised semantic segmentation pipeline.

Source: produced by the author.

This comprises the primary pipeline, as initially proposed. In this first work [Kanezaki, 2018], the pipeline was supposed to be trained and used for single images individually or maybe for similar images, such as near frames in a video. But this pipeline can achieve much better results, and, for example, with a proper feature extractor, the semantics of an entire dataset could be learned. To explore the pipeline, we further discuss each block's possible optimizations.

3.1.1 Image Preprocessing

The preprocessing step is usually very straightforward. Generally, each standard network in literature, such as VGGNet, ResNet, and others, has its preprocessing layer. That is, their own required image size and type of normalization. Using a proper image size and a normalization technique is valid; the first is to prevent large memory requirements, and the latter is to ensure that the network weights are not too high or too low, preventing the gradient from vanishing or exploding.

These valuable techniques are generally used, but another critical preprocessing step is the data augmentation [Shorten and Khoshgoftaar, 2019]. This technique performs several transformations in the images during the training step, such as rotations, translations, color changes, etc. These transformations will help the network filters generalize even further, allowing the results to become more consistent and adding more variety and complexity to the semantics the network learns.

The problem with data augmentation concerning semantic segmentation is that some techniques must also be applied to the semantic mask, limiting the possibilities for usage. Furthermore, since the goal of this pipeline is also to apply weak supervision later, this also must be considered when selecting the data augmentation techniques. The only work derived from Kanezaki [2018] that used some data augmentation was the work of Lin et al. [2020], which applied horizontal flips and random crops of 300×300 pixels. In this work, the following data augmentation operations were performed:

- Random color jitter: random variations in brightness, contrast, saturation, and hue.
- Random Gaussian blur: variations in kernel size and standard deviation.
- Random affine transformation: random variations in scale, shear, translation, and rotation.
- Random horizontal and vertical flips

With these techniques, we tried to provide the highest amount of generalization to the images, thus elevating the potential and complexity of the semantics learned by the network.

3.1.2 Feature Extractor

The feature extractor is perhaps the most crucial block in this pipeline since it must be capable of generalizing for different semantic contexts, allowing the network to thoroughly learn the visual features of many datasets. In the original work of Kanezaki [2018], the authors used a combination of M simple feature extraction blocks, where each block was composed of a 2D convolutional layer with filters of size 3×3; a ReLU activation layer; and a Batch Normalization layer. In Kanezaki's, the authors tested $M = 1, 2, 3$ and achieved the best result with $M = 2$, but in a posterior extension of the work [Kim et al., 2020], they fixed $M = 3$.

Other authors tried different feature extractors in posterior works that derived from this previous one. Kim and Ye [2020] used the U-Net to perform the feature extraction. Saha et al. [2019] used a combination of transfer learning, from which they used VGG-16 trained in ImageNet, and learnable layers, where they used 4 of the basic original feature extraction blocks. Barthakur and Sarma [2019] used SegNet to extract features. Bhatt et al. [2019] used an FCN. Saha et al. [2020] used only the original feature extraction blocks, with $M = 4$ and $M = 5$. Ilyas et al. [2020] used a variation of the original feature extraction block, called SE-Block, where basically, they exchanged the convolutional layer of the original block to a new block with a "squeeze and exciting" scheme of layers, where 1×1 convolutions are applied to decrease the number of features, and after a few 1×1 layers, the output returns to the original size, similar to what is performed in MobileNet; more information about this block can be found in [Hu et al., 2020]. Khan and Yang [2020] propose a new variation of the U-Net, which they call FCD u-net. It benefits from feature reuse, where the feature maps from previous layers are concatenated, increasing the variation of the subsequent layers' inputs. Zhou and Wei [2020] used an encoder-decoder architecture using 6 of the original feature extraction blocks, where there are two blocks, a max pooling layer to decrease the size, two more blocks, a deconvolutional layer, the last two blocks, and a simple convolutional layer. And finally, Lin et al. [2020] used a Superpixel Autoencoder, or SuperAE, an encoder-decoder architecture composed of 6 of the original feature extraction block, three for the encoder and three for the decoder.

As seen, many different strategies have been employed to solve this problem. Overall, several interesting feature extractor variations could potentially present excellent results. In this work, we compare three feature extractors, from which we try to compare which one has the highest ability to explain the semantics of a dataset. For that, we test several of these already-used feature extractors. More details regarding our choice can be seen in the experiments section.

3.1.3 Pixel Classifier

It seems straightforward about the pixel classifier: a softmax classifier. Kanezaki [2018] used a simple Softmax Loss, or Cross Entropy Loss, to classify the pixels and propagate the results, but other works proposed different approaches. In their extension work [Kim et al., 2020], the authors used a combination of two losses, the common cross-entropy loss, based on the assigned labels to enhance feature similarity, and a spatial continuity loss, that consists in summing the L1-norm of horizontal and vertical differences of the response map. Optimizing the second loss means fewer label differences, thus suppressing complicated patterns or textures with too many possible labels.

Other authors used a variety of other losses to optimize the classifier. Kim and Ye [2020] used the Mumford-Shah Loss, which works by minimizing the variational energy of the image segmentation, forcing each segment to have similar pixel values. Saha et al. [2020] worked with multi-temporal images, so they proposed a loss used to detect object segments from individual images and establish a correspondence between distinct multitemporal segments, which sums the cross entropy loss for multiple images over time. Finally, Lin et al. [2020] proposed the usage of both the Cross-Entropy Loss and another loss that takes into account the similarity between superpixels. Their idea is that superpixels similar in deep features must have a higher probability of belonging to the same class, so they build a similarity matrix between the superpixels and model this optimization problem, minimizing it during training.

Besides the original loss, each of the other losses was proposed aiming at some improvement. In our work, we proposed to use a variation of the actual Cross Entropy Loss that considers the class weights, giving more importance to classes that appear less in the dataset. We also try to add a few of the previously proposed losses to check if they help to improve semantics learned by the network. This last experiment will be better described in the Results chapter. As for our proposed loss, we begin the explanation with the softmax equation.

Consider the simple case of logistic regression. In this problem you have two outputs, either 0 or 1, and you first calculate the linear product of z, given by:

$$z = \vec{w} \cdot \vec{x} + b$$

where $\vec{w}$ are the weights, $\vec{x}$ is the input vector with multiple features, and b is the bias term of the current layer.

Then you calculate the activation $a = g(z)$, which is the sigmoid function applied to z:

$$a = g(z) = \frac{1}{1 + e^{-z}} = P(y = 1|\vec{x})$$

This can be interpreted as the logistic regression estimate of the probability of y being equal to 1 given the input features $\vec{x}$.

If we expand the sigmoid equation, we have:

$$\frac{1}{1 + e^{-z}} = \frac{1}{1 + \frac{1}{e^z}} = \frac{1}{\frac{e^z+1}{e^z}} = \frac{e^z}{e^z + 1}$$

Now, if the probability of $y = 1$ is 0.7, the likelihood of $y = 0$ must be 0.3. If we consider it, the logistic regression gives us two probabilities for two classes: one or zero, or a_1 and a_2.

If we interpret $a_1 = 1 - a_2$, we can interpret the output as the percentage of a_1 given two classes, a_1 and a_2:

$$\frac{a_1}{a_1 + a_2} = \frac{\frac{1}{1+e^{-z}}}{\frac{1}{1+e^{-z}} + 1 - \frac{1}{1+e^{-z}}} = \frac{\frac{e^z}{e^z+1}}{1} = \frac{e^z}{e^z + 1}$$

The softmax is a way of generalizing this methodology by taking the exponential of each activation and dividing it by the different activations. In other words, you normalize each activation by the sum of all activations. Given the output for a given class j, in a problem with N classes

$$z_j = \vec{w_j} \cdot \vec{x} + b_j, j = 1, \ldots, N$$

The softmax activation would be

$$a_j = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}} = P(y = j|\vec{x}) \tag{3.1}$$

We can think of the softmax as a layer to be included in a neural network. Still, instead of being a single neuron, each class in the softmax will be a different node, where each node will perform a different logistic regression for a given class. The complete result **a** for multiple classes is a vector of length N, which can be written as

$$\mathbf{a}(x) = \begin{bmatrix} P(y = 1|\mathbf{x}; \mathbf{w}, b) \\ \vdots \\ P(y = N|\mathbf{x}; \mathbf{w}, b) \end{bmatrix} = \frac{1}{\sum_{k=1}^{N} e^{z_k}} \begin{bmatrix} e^{z_1} \\ \vdots \\ e^{z_N} \end{bmatrix} \tag{3.2}$$

In softmax regression and neural networks with softmax outputs, N outputs are generated, and one output is selected as the predicted category. In both cases, a vector **z** is outputted by a linear function applied to a softmax function. The softmax function converts **z** into a probability distribution. After applying softmax, each output will be between 0 and 1, and the outputs will be added to 1, allowing them to be interpreted as probabilities. The more significant Z inputs will correspond to larger output probabilities.

As for the actual loss, consider the cost function for the logistic regression. Given that the resulting activation is a_1, that is, $P(y = 1|\vec{x})$, and that the activation $a_2 = 1 - a_1$, that is, $P(y = 0|\vec{x})$, the logistic regression loss is given by

$$loss = -y\ log\ a_1 - (y-1)\ log\ (1-a_1) = -y\ log\ a_1 - (1-y)\ log\ a_2$$

which is the same as $-log\ a_1$ if $y = 1$ and $-log\ a_2$, if $y = 0$.

For the softmax regression, we can generalize that by saying that the loss will be

$$L(\mathbf{a}, y) = loss(a_1, \ldots, a_N, y) = \begin{cases} -log(a_1), & \text{if } y = 1 \\ -log(a_2), & \text{if } y = 2 \\ & \vdots \\ -log(a_N), & \text{if } y = N \end{cases} \tag{3.3}$$

where y is the target category for this example and $\mathbf{a}$ is the output of a softmax function. In particular, the values in $\mathbf{a}$ are probabilities that sum to one.

Note in the equation above that only the line corresponding to the actual target contributes to the loss, while the other lines are zero. To write the cost equation, we need an "indicator function" that will be 1 when the index matches the target and zero otherwise.

$$\mathbf{1}\{y == n\} = \begin{cases} 1, & \text{if } y == n. \\ 0, & \text{otherwise.} \end{cases}$$

Now the cost can be defined as

$$J(\mathbf{w}, b) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{N} \mathbf{1}\left\{y^{(i)} == j\right\} \log \frac{e^{z_j^{(i)}}}{\sum_{k=1}^{N} e^{z_k^{(i)}}}\right] \tag{4}$$

Where m is the number of examples, N is the number of outputs. This is the average of all the losses.

This Softmax Loss is also called Cross Entropy Loss. And finally, for the Weighted Cross Entropy Loss, we also multiply the normalized class weights c_j for each of the N classes. The equation is given by

$$WCELoss = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{N} \mathbf{1}\left\{y^{(i)} == j\right\} c_{j_{norm}} \log \frac{e^{z_j^{(i)}}}{\sum_{k=1}^{N} e^{z_k^{(i)}}}\right] \tag{4}$$

Notice that since we are solving an Unsupervised problem, we don't have access to the class weights beforehand, which differs from a supervised or weakly supervised problem. To include the class weights, we propose calculating them by batch. First, we take every image of a batch and feedforward through the network as usual. Then we use the superpixel refinement step to find the classes for each image. Finally, we use these labels to calculate the class weights considering only the classes present in the current batch. For example, consider a case where we initialize our network with $q = 100$ classes, and only ten different classes appear in the given batch. The weights of the remaining classes will be zero, while the weight of a class j between the n present classes will be

$$c_j = \frac{N_{\text{pixels}}}{\text{freq}_j * N} \tag{3.4}$$

where freq_j indicates the frequency of the current class in the current batch, or the number of pixels belonging to this class, and N_{pixels} is the total number of pixels in the current batch.

This will generate a vector with one weight for each class, and classes with more pixels present will have lower weights. Finally, the weights are normalized so they don't become values too large:

$$c_{j_{norm}} = \frac{c_j}{\sum_{j=1}^{N} c_j} \tag{3.5}$$

3.1.4 Superpixel Refinement

The final step of the pipeline is the superpixel refinement step. The original author proposed the usage of the Slic algorithm [Achanta et al., 2012] to perform a superpixel segmentation, where the class of the pixels, chosen by the "arg max" function, inside each superpixel, would go through a majority vote process to define the entire superpixel class, which would later be used as the "correct result" for backpropagation. In their extension work [Kim and Ye, 2020], they completely removed the refinement step by just comparing the network output with the index of the highest activation (in other words, the cluster label). They achieve the optimal semantic segmentation by only optimizing their losses, thus not needing this refinement.

As for the other derived works, some chose different approaches. Bhatt et al. [2019] uses a strategy similar to W-Net, by first using the Normalized Cut algorithm to generate the superpixels. Then, after the feedforward step, they apply post-processing to the normalized response maps of the network using Conditional Random Fields, whose purpose is to smooth and refine the output. Finally, the superpixel refinement is performed as usual. Khan and Yang [2020] uses the standard Slic segmentation and refinement but afterward performs a region-merging process by extracting 44 different features from each superpixel and constructing a Region Adjacency Graph, leading to a Region Merging Algorithm based on the graph distance matrix. Among the features extracted from the superpixels are spatial, shape, texture, color, and color histogram features. This post-processing step help to reduce the over-segmentation of the images, a common problem when using Slic. Lin et al. [2020] also uses Slic, but their work uses an encoder-decoder architecture, where the image is reconstructed. The authors use Slic in this reconstructed image, justifying that this image is smoothed, therefore generating superpixels that will pay attention to more meaningful regions.

In our work, we believe that the features used by Slic and the algorithm do not provide an optimal base segmentation. Besides, since the network is learning to perform the semantic segmentation from scratch, and specifically in the case of remote sensing imagery, the training can be expensive in both time and computational power, and using poor segmentation as ground truth can lead to a more extended training and a worse semantic generalization. Therefore, instead of using superpixels to combine the classes, it might be more beneficial to use a more robust methodology to obtain the proposed regions to refine the network clusters, or classes, such as an unsupervised semantic segmentation algorithm that does not require training and that preferably already have a good performance in remote sensing tasks. By using an external algorithm, it is possible to extend the unsupervised semantic segmentation pipeline [Kanezaki, 2018] by exchanging the superpixel refinement step for a class refinement step provided by any semantic segmentation. This can be illustrated in Figure 3.

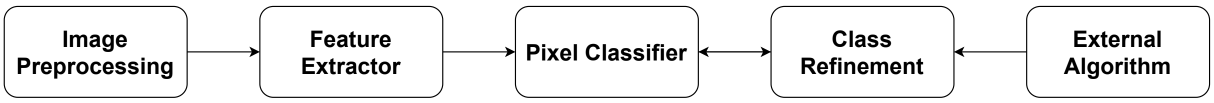


Figure 3 – Modified unsupervised semantic segmentation pipeline.

Source: produced by the author.

The main advantage of this proposed methodology is that the external segmentation can be provided by either an unsupervised algorithm or perhaps it could be the ground truth itself, turning the problem into a supervised approach.

As for our choice, we use the unsupervised semantic segmentation pipeline proposed by [Jaimes et al., 2022]. As illustrated in Figure 4, this algorithm receives an input image, applies bilateral smoothing filtering, performs a superpixel segmentation using Slic, extracts 345 color and texture features from each superpixel, and finally performs a series of post-processing clustering steps: define the ideal number of clusters, achieve the optimal clustering of the superpixels based in the Calinski-Harabasz Index, and use a Hierarchical Clustering technique to group similar clusters, then obtaining the final semantic mask.

The pipeline from Jaimes et al. [2022] groups ideas from the other works that proposed enhanced refinements, such as creating superpixels from a smoothed image [Lin et al., 2020] and merging regions based on features extracted from the superpixels [Lin et al., 2020]. Therefore, we consider this pipeline robust and hypothesize that it will

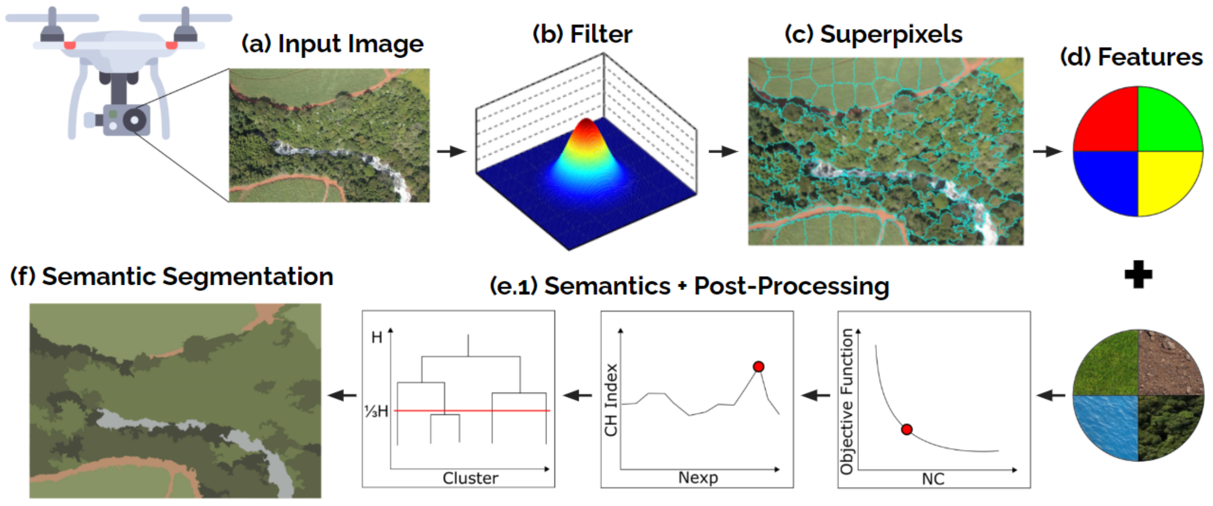


Figure 4 – Handcraft unsupervised semantic segmentation algorithm.

Source: adapted from Jaimes et al. [2022].

provide better ground truth for the refinement step than raw superpixels since it provides preliminary information on the superpixel clusters to the network.

Finally, our pipeline can work with unsupervised and supervised inputs, but it can also be extended to work with weakly supervised labels. By exchanging the ground truth signals from masks that comprehend the entire image area to weak labels, the same logic, i.e., adopting the majority class of the pixels inside a scribble or polygon as the ground truth class, the pipeline can receive all kinds of inputs. The minor adaptations are setting the network to recognize an unknown class that will weigh 0 during backpropagation and not considering this class when calculating class weights.

With this last block properly analyzed, we conclude the pipeline for unsupervised semantic segmentation. The pipeline is categorized as general because it accepts multiple types of supervision, can consider class weights within training, can be coupled with various losses besides the Cross-Entropy loss, and can provide an efficient semantic mapping of the characteristics of given images, learning clusters that represent most of their content.

3.2 Weak Dataset Labeling

The idea for an automatic scribble generator is based on works that focus on searching for similar regions in entire datasets Batra et al. [2011], Hamilton et al. [2022] and works that can generate automatic scribbles Ding et al. [2012]. In remote sensing images, there are usually well-defined crops and types of terrain, especially in rural areas. Based on that hypothesis, if such regions exist, it should be possible to group them based on the similarity of region features (descriptors). Differently from a well-defined semantic segmentation pipeline, the idea here is not to describe the entire image, only homogeneous areas, in the same way that perhaps a human would unintentionally draw scribbles, given the clarity of the region.

This section will be divided into two parts; the first is how we managed to create an automatic scribble generator, and the next is how we extended this idea to label entire datasets with minimal cost.

Note that we also published a paper with part of our experiments [Ferreira et al., 2022], where more information regarding this pipeline can be found.

3.2.1 Automatic Scribble Generation

Differently from a well-defined semantic segmentation pipeline, the goal is not to describe the entire image, only large homogeneous areas of it, the same areas that perhaps a human would unintentionally draw scribbles given the clarity of the region. Before explaining our proposed pipeline, we first analyze a previous work in the literature.

In an application for image colorization through scribble labeling, Ding et al. [2012] proposed an automatic scribble generation methodology. First, the image is segmented using a Graph-Based Image Segmentation algorithm. Then, starting at the center pixel, they perform an 8-direction search for areas with high information density, calculated using a Spatial Distribution Entropy technique. The process is repeated until finding a sufficient number of points, and finally, fitting a smoothed curve to them generates a scribble.

There are a few limitations to this approach. The first focuses on grayscale images, whereas ours focuses on RGB images. The second limitation is that every segmented region receives a scribble, sometimes creating an overpopulation of scribbles. Since this is a coloring application, and humans are required to label these scribbles, having too many scribbles could be efficient in this particular case, but not for remote sensing imagery labeling, where we propose to label several thousands of images.

Therefore, we propose a pipeline to tackle these problems and generate scribbles, focusing on labeling remote sensing datasets. The complete pipeline can be seen in Figure 5. Overall, the main steps are similar to Ding et al. [2012] but performed differently. Each stage will be subsequently discussed.

Starting with a remote sensing input image in step 1.1, the first goal is to segment the image into different regions, so we can search for places to draw the scribbles. But first, we must find the number of superpixels in which we shall segment the image, which varies between different images. Some images with homogeneous content with a single semantic object in the entire image, such as pictures of a dense forest or an expansive grass field, don't need to be segmented into many regions because, in this case, a single scribble could describe the entire image. On the contrary, images with different areas and lots of artifacts, such as houses, must be segmented into more superpixels so that smaller

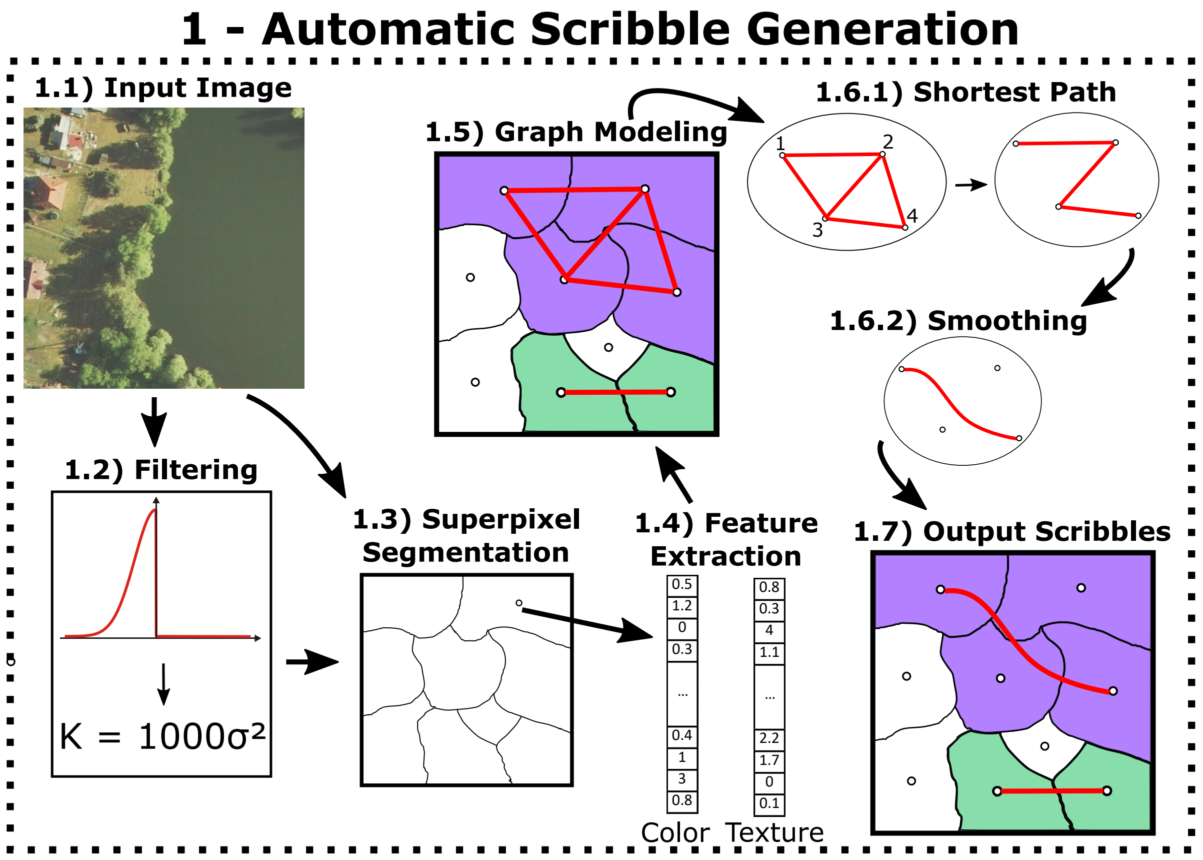


Figure 5 – Weak label generation pipeline. Given an input image (1.1), image filtering (1.2) and superpixel segmentation (1.3) are performed, followed by a superpixel feature extraction (1.4), used to model a graph (1.5), where similar neighbor superpixels are found, and a shortest path algorithm (1.6.1) creates a path between their centers, that can be smoothed (1.6.2) to generate scribbles between them (1.7).

Source: Ferreira et al. [2022].

regions can also be described.

To deal with this problem, we can describe this difference in the image contents using its pixels variance value. This way, images with a lower value will have fewer artifacts than images with a higher value. But if the images have complex textures, such as detailed grass, the variance would still be high, even for simple images. Therefore, we propose to apply a bilateral filter in step 1.2 to homogenize the contents of the images, with the benefit of preserving the edges. We then obtain the variance of the pixels for the smoothed image, $ImVar$.

Then, in step 1.3, the image is segmented into $1000 * ImVar$ superpixels using the Slic algorithm, where 1000 is a constant scale factor to balance the usage of normalized images, which varies between 0 and 1, making the variance a small number. Notice that we will extract features from these superpixels later, so while the smoothed image allows a better estimate of the image variance, it can lead to a loss of information regarding the image content. To avoid that, we use the original image from this step forward, meaning that the filtered image is only used to calculate the variance.

Next, in step 1.4, we extract 375 features from the superpixels as described in Jaimes et al. [2022], being 105 color (f_c) features extracted from seven different color spaces, and 270 texture (f_t) features, derived from a Leung-Malik filter bank. A graph is mounted in step 1.5 using adjacent superpixels, with the weights, ϕ_{ij}, adapted from Lin et al. [2016]:

$$\phi_{ij} = exp\left\{ -\frac{\|f_c(x_i) - f_c(x_j)\|_2^2}{\delta_c^2} - \frac{\|f_t(x_i) - f_t(x_j)\|_2^2}{\delta_t^2} \right\} \tag{3.6}$$

where δ_c is set to 0.5 and δ_t to 2. Differently from Lin et al. [2016], here we use different color and texture features, and the weights are calculated using the features themselves, not their histograms.

Assuming that similar superpixels will have a closer distance in the feature space, which varies from 0 to 1, we set adjacent superpixels to the same region if their edge weight is higher than 0.9, that is, they have 90% of similarity according to the metric in Equation 3.6. A graph search is performed, and all groups are mounted, representing the regions of similarity in the image, and each group will be used to generate a scribble. In images with a higher variance and several superpixels, many groups are prone to be formed, leading to an overpopulation of scribbles. To deal with this problem, we set the minimum number of superpixels in a valid group that will receive a scribble to 5% of the number of superpixels of the current image, avoiding noisy small groups. Finally, the scribbles may be drawn via the adjacent superpixels in each group, representing the regions of similarity in the image.

To draw the scribbles in step 1.6.1, the center of each superpixel in a group is first taken, and the total height and width are determined. If the height is higher than the width, the highest center of the superpixels will be the first in the scribble; otherwise, it will be the leftmost one. After defining the initial center, the shortest path among the centers is found through a generic solver for the traveling salesman problem. The problem is modeled so the salesman does not have to return to the original center; it must only find the fastest way throughout the centers while going over every center once. For a large number of points solving this problem could take a long time, but for a small number of points, as is expected in this application, the solution can be found fast.

Approximating the shape of human-generated scribbles can be beneficial in cases where the superpixels form a curved shape, although in most cases, only the visual is improved. Because of that, in optional step 1.6.2, after a bicubic interpolation to increase the number of points, a Gaussian filter is applied to smooth the scribble, with a window size of half the number of points, achieving the final result in step 1.7, where the scribbles are stored in external files. The process for obtaining the scribbles classes will be described in the next section.

One problem observed with this methodology was the difficulty of segmenting small objects into more than one superpixel, such as buildings in a largely rural area, causing them not to reach the minimum amount to originate a scribble. Therefore, as a solution to this problem, we propose to use a separate detector that can be coupled with our pipeline. We tested adding a building detector [Sirmacek and Unsalan, 2011] to the pipeline, supposing a dataset with buildings as a class to improve their representativity, forcing superpixels with any facilities to become scribbles. As the authors suggested Sirmacek and Unsalan [2011], some parameters that vary for each dataset were adjusted. To use this resource more efficiently, we assume that images with buildings will likely have a higher pixel variance, given the different colors and shapes. By calculating the variance of the entire dataset previously, as an offline step, we take the highest variance in the dataset and only apply the building detection in images with at least 25% of the highest variance, which in this case corresponds to around 15% of the dataset. This way, most images that are not likely to have buildings can be filtered.

3.2.2 Scribble Classification

After automatically labeling scribbles, they still don't have an associated class, so we have several unlabeled scribbles. The straightforward solution would be to manually allocate labels to every scribble, meaning that the only purpose of the automatic labeling process would be to save time for drawing the scribbles. But since our goal is to provide a way of automatically labeling most labels, our first hypothesis is to prove if it is possible to find patterns among all images in a given dataset so that the same classes can be found in multiple images.

For that, we analyze two different works. The first is related to Interactive Co-Segmentation [Batra et al., 2011], to separate foreground objects from their background, where the authors propose to solve this problem using intelligent scribble guidance. To alleviate the scribble drawing process, the authors suggest "iCoseg," an automatic recommendation system that intelligently indicates where the user should draw the following scribbles in a group of images, given that the user has drawn scribbles in one or more images. They then segment the images into superpixels and extract seven cues, or features, for each superpixel, which then is fed to a logistic regression classifier that will learn from the images with scribbles and classify the remaining, generating a set of regions with the potential to receive new scribbles from the user.

The second work performs unsupervised semantic segmentation through visual transformers [Hamilton et al., 2022]. We are interested in the first step of this methodology, where they extract patches from several images in the dataset and perform self-supervised feature learning. When passing these patches through the same backbone, a segmentation head is trained with a loss function that maximizes feature similarity of patches from the same image while minimizing feature similarity between patches from different images. In the end, similar regions of similar images will have closer feature vectors, and this can be used to verify if two patches are from a similar class.

Both these works prove, with different approaches, that it is possible to extract meaningful features among images in the same dataset, allowing the features categorization into classes, therefore being possible to learn to represent such classes. Inspired by this idea, we propose using a classifier to learn features from a small set of previously labeled scribbles that will be able to label several unseen scribbles automatically. The proposed pipeline can be seen in Figure 6.

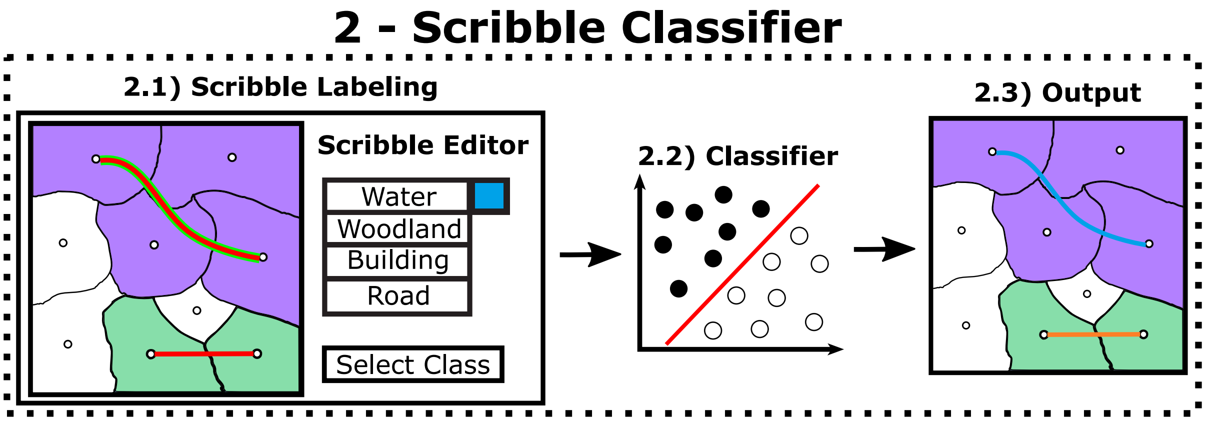
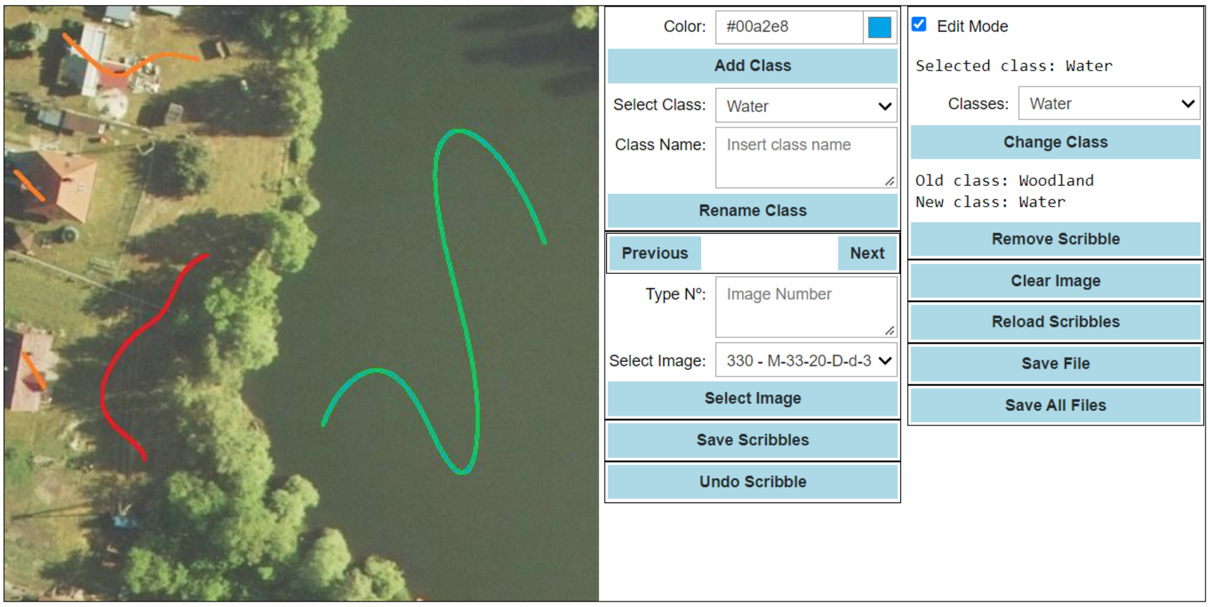


Figure 6 – Scribble classifier pipeline. Given several images with unlabeled scribbles, a small set of images are labeled with our provided tool (2.1), which is used to train a classifier (2.2) and finally automatically label the remaining images (2.3).

Source: Ferreira et al. [2022].

Starting with the unlabeled, automatically generated scribbles in step 2.1, we provide a graphical tool for labeling them, which allows drawing, editing, removing, and changing the class (annotating) of already drawn scribbles with a few mouse clicks is possible. Note that when automatically generating the scribbles, some may be too small or have a bad shape, not correctly representing the feature they should label. To deal with this issue, with our tool, the user can auto-generate the scribbles, label them, and fix the worst ones by removing them or manually drawing better representations. The graphical user interface can be seen in Figure 7.

After labeling a set of images with at least a few scribbles of all classes in the dataset, we propose to use an SVM to auto-classify the remaining images in step 2.3. We use their already extracted color and texture features as input for each superpixel, as described in Section 3.2.1. Since one scribble comprises multiple superpixels, we take the mean of their features, weighted by their areas. And finally, the remaining scribbles are appropriately classified in step 2.3.



Color: #00a2e8

Select Class: Water

Select Image: 330 - M-33-20-D-d-3

[✓] Edit Mode

```
Selected class: Water
```

Classes: Water

```
Old class: Woodland
New class: Water
```

Figure 7 – An overview of our scribble editing tool, a graphical user interface to visualize, draw, edit, remove, and manually annotate scribbles.

Source: produced by the author.

This pipeline works for an auto-generated dataset of scribbles. However, if the user wants to use an already manually drawn dataset, the pipeline is still applicable, with some minor modifications from the pipeline in Figure 3.2.1. By segmenting the image into superpixels, extracting their features, and checking which superpixels are crossed by each scribble, it is possible to calculate the mean features and recreate the scribble dataset from any image. This roughly corresponds to steps 1.1 to 1.4.

This concludes this section and chapter. In the next chapter, we shall discuss the several experiments performed to test and analyze our proposed methodologies. We will also present a few remote sensing applications in which our pipelines can be applied.

Chapter 4

Experimental Setup

In this chapter, we define the experimental setup for our experiments, including the variety of metrics that we used to evaluate the performance of our algorithms, how we mapped our classes from an unsupervised methodology to a supervised result, and the datasets that we used in our work for each experiment.

As for the machines, we used two computers to perform the experiments, one composed of an i9-9900KF processor, 32GB DDR4 RAM, and an RTX-3060 12GB graphics card, and the other with an i7-9750H processor, 32GB DDR4 RAM, and a GTX-1660 Ti 6GB graphics card.

4.1 Metrics

In our experiments, we evaluate different metrics for different purposes. Some metrics are quantitative and used to assess the actual performance of our model in a supervised manner. In contrast, other qualitative metrics evaluate if the obtained segmentation provides meaningful information.

4.1.1 Supervised Quantitative

Starting with the supervised quantitative metrics, before defining the metrics, we may first define some concepts. Given a set of pixels, the predicted class is the one the classifier assigns, and the target class is the real ground truth label of the pixel. Therefore:

- a True Positive (TP) is predicted positive for a given class ($\hat{y} = 1$), and the target label is also positive for the same class ($y = 1$).
- a True Negative (TN) is predicted negative for a given class ($\hat{y} = 0$), and the target label is also negative for the same class ($y = 0$).

- a False Positive (TN) is predicted positive for a given class ($\hat{y} = 1$), but the target label is negative for the same class ($y = 0$).
- a False Negative (TN) is predicted negative for a given class ($\hat{y} = 0$), but the target label is positive for the same class ($y = 1$).

The first and most common metric is Accuracy (Acc). This metric measures the total number of TPs and TNs divided by the total number of pixels:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

The Precision (P) is a metric that measures the proportion of correct positive predictions. In other words, this indicates how many classified pixels are valid.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

The Recall (R), or Sensitivity, or True Positive Rate, is a metric that measures the proportion of actual positive predictions that were correctly identified. In other words, this indicates how many correct pixels were classified.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

The Specificity (S), or Selectivity, or True Negative Rate, is a metric that measures the proportion of actual negatives that was correctly identified. In other words, this indicates how many incorrect pixels were classified.

$$Specificity = \frac{TN}{FP + TN} \tag{4.4}$$

The F1-Score metric, or Dice Coefficient, is a harmonic mean of both the Precision and the Recall. Optimizing this metric means finding the best balance between FPs and FNs. And another way of thinking about this metric is the ratio between the area of intersection between two segmentations, where they are equal, divided by the total number of pixels of both images.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{4.5}$$

The Intersection over Union (IoU) metric, or Jaccard Index, is given by the ratio between the area of overlap between two segmentations, where they are equal, divided by the union between the predicted segmentation and the ground truth. This is similar to the

F1-Score, but the F1 considers the overlap area twice, once in each segmentation, whereas the IoU considers it only once.

$$IoU = \frac{TP}{TP + FP + FN} \tag{4.6}$$

These metrics so far are calculated individually for each class. So in a multiclass problem, this would result in several IoU calculations, one for each class. So, given C classes, the Mean IoU (mIoU) is the mean of the IoU for each class:

$$mIoU = \frac{1}{C}\sum_{c=1}^{C} IoU_c \tag{4.7}$$

This leads directly to the next metric, which is the Weighted Mean IoU (WmIoU), or the Mean IoU weighed by the contribution of each class. This weight can have multiple meanings, but it usually is the number of pixels of a given class in the ground truth divided by the total number of pixels.

$$WmIoU = \frac{1}{C}\sum_{c=1}^{C} w_c \cdot IoU_c \tag{4.8}$$

4.1.2 Segmentation Quantitative

The segmentation quantitative metrics are widely used to analyze the segmentation quality as a clustering problem. We start with the Variation of Information (VI) metric [Meilă, 2005]. This is usually used to evaluate clustering in general, but that has also been widely used to assess unsupervised semantic segmentation [Xia and Kulis, 2017, Kim and Ye, 2020, Ilyas et al., 2020, Khan and Yang, 2020, Zhou and Wei, 2020, Lin et al., 2020]. It measures the distance between two segmentations regarding their average conditional entropy. In other words, it roughly measures the amount of randomness in one segmentation, which cannot be explained by the other. The equation is given by

$$VI(S_1, S_2) = H(S_1) + H(S_2) - 2I(S_1, S_2) \tag{4.9}$$

where $H(S)$ is the entropy of a given segmentation S, and $I(S_1, S_2)$ is the mutual information between segmentations S_1 and S_2.

In a practical perspective, given that a segmentation X is composed of several subsets, $X = \{X_1, X_2, \ldots, X_k\}$, where each X_i is a set of pixels composing a given partition of the complete segmentation. Equally, a second segmentation Y is $Y = \{Y_1, Y_2, \ldots, Y_l\}$ And suppose that $|X_i|$ is the number of elements inside a subset. We can define the total number of elements as

$$n = \sum_i |X_i|$$

Now consider the proportion of a single subset relative to the others as

$$p_i = \frac{|X_i|}{n} \text{ and } q_j = \frac{|Y_j|}{n}$$

And finally, consider the number of elements in common between a subset of both segmentations as $|X_i \cap Y_j|$. Its proportion related to the total is given by

$$r_{ij} = \frac{|X_i \cap Y_j|}{n}$$

The variation of information is then given by

$$VI(X,Y) = -\sum_{i,j} r_{ij} \left[log\left(\frac{r_{ij}}{p_i}\right) + log\left(\frac{r_{ij}}{q_j}\right)\right] \tag{4.10}$$

Larger values of this metric correspond to the more significant dissimilarity between the clusterings; therefore, lower VI values mean better segmentation results.

Another metric is the Probability Rand Index (PRI) [Pantofaru and Hebert, 2005], also used by a number of works [Xia and Kulis, 2017, Kim and Ye, 2020, Khan and Yang, 2020, Zhou and Wei, 2020, Lin et al., 2020]. This metric counts the fraction of pairs of pixels whose labelings are consistent between the computed segmentation and the ground truth, averaging across multiple ground truth segmentations. In other words, it measures the similarity between two data clusters.

The Rand Index can be understood as the number of agreements between two segmentations divided by the number of agreements summed with the number of disagreements. Given a set of n elements $S = o_1, \ldots, o_n$, and two partitions of S to compare, $X = X_1, \ldots, X_r$, a partition of S into r subsets, and $Y = Y_1, \ldots, Y_s$, a partition of S into s subsets, consider:

- a is the number of pairs of elements in S that are in the same subset in X and Y
- b is the number of pairs of elements in S that are in different subsets in X and Y
- c is the number of pairs of elements in S that are in the same subset in X and different subsets in Y
- d is the number of pairs of elements in S that are in different subsets in X and in the same subset in Y

The Rand Index is calculated by

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\frac{n}{2}} \tag{4.11}$$

where $a+b$ is the number of agreements between X and Y and $c+d$ is the number of disagreements between X and Y.

The Probabilistic Rand Index is the Rand Index averaged by every image, and a higher PRI value indicates a more accurate segmentation.

The Global Consistency Error (GCE) metric [Martin et al., 2001] is also used by some works [Khan and Yang, 2020, Zhou and Wei, 2020]. It measures the extent to which one segmentation can be viewed as a refinement of another. Segmentations related in this manner are considered consistent since they could represent the same natural image segmented at different scales.

Given that segmentation is a division of the pixels of an image into sets and that the segments are sets of pixels, if one segment is a proper subset of the other, then the pixel lies in an area of refinement, and the error should be zero. If there is no subset relationship, then the two regions overlap inconsistently.

Given two segmentations of the same size, S_1 and S_2, consider the measure of the error at each pixel p_i

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \setminus R(S_2, p_i)|}{|R(S_1, p_i)|} \tag{4.12}$$

where $R(S_j, p_i)$ is the region in segmentation j that contains pixel p_i, $\setminus$ denotes set difference, and $|\cdot|$ denotes set cardinality. This measure evaluates to 0 if all the pixels in S_1 are also contained in S_2, thus achieving the tolerance to refinement discussed above. Given the error measures at each pixel, the error between two segmentations is defined as

$$GCE = \frac{1}{n} \min \left\{ \sum_i E(S_1, S_2, p_i), \sum_i E(S_2, S_1, p_i) \right\} \tag{4.13}$$

For this metric, lower values mean closer segmentations.

Another metric is the Boundary Displacement Error (BDE) [Freixenet et al., 2002], also used to compare semantic segmentations [Xia and Kulis, 2017, Khan and Yang, 2020, Zhou and Wei, 2020]. It measures the average displacement error of boundary pixels between two segmented images. In other words, it defines the error of one boundary pixel as the distance between the pixel and the closest pixel in the other boundary image.

Given the boundaries of two segmentations, obtained by taking the absolute values of their gradients and checking for values above zero, a bipartite graph matching problem is solved by first constructing the distance matrix between the boundary elements in each image, associating with the nearest pixel in the other boundary. Then, the percentage of matched edge elements is computed and taken as the boundary error. This error is calculated for both segmentations, and its mean will be the BDE value. The lower it is, the closer both segmentations are.

Finally, the last metric is the Segmentation Covering (SC) metric [Arbeláez et al., 2011], also used by some works to compare segmentations [Zhou and Wei, 2020, Lin et al., 2020]. It measures the overlap of regions from the segmentation output and the ground truth.

The overlap between two regions R and R' is given by

$$\mathcal{O}(R, R') = \frac{|R \cap R|}{|R \cup R|}$$

where $|\cdot|$ is the area; in other words, it is the intersection over the union. The covering of a segmentation S by another segmentation S' is:

$$SC = \mathcal{C}(S \to S') = \frac{1}{N}|R| \cdot \max_{R' \in S'} \mathcal{O}(R, R') \tag{4.14}$$

where N denotes the total number of pixels in the image. In other words, the area of every segment in S is multiplied by the highest overlap with the segments in S'. The sum for every segment is taken and pondered by the number of pixels. The higher the SC value, the better the quality of segmentation.

4.1.3 Metrics Summary

We comprise all metrics in Table 1 to facilitate the reading, organization, and posterior consultation. The table is composed of the name of the metrics, a brief description of what the metrics try to measure, and how to analyze the metric, where " $\uparrow$ " indicates that the higher the metric value is, the better is the segmentation result, and similarly, for " $\downarrow$ " the lower the metric value, the better is the segmentation.

4.2 Class Mapping

It is essential to notice that since our approach is unsupervised, becoming a clustering problem, our cluster classes have no direct association with supervised classes. To overcome this problem, we perform a two-step evaluation. First, we feedforward every image in the test dataset through our network and store the corresponding pixel activations of each of our K clusters with each dataset's original classes by comparing values individually, resulting in a histogram table for each class. For example, cluster 3 may

Table 1 – Metrics used to evaluate semantic segmentation in this work.

Metric	**What it measures**	**How to analyze**
Accuracy	The percentage of correctly assigned pixels in a predicted segmentation	↑
IoU	The overlap between a target and predicted segmentation	↑
WIoU	The same as IoU, but weighted by the number of pixels in each class	↑
F1-Score	The correct pixels considering the number of wrong positives and wrong negatives	↑
VoI	The distance between the two segmentations in terms of their average conditional entropy	↓
PRI	The accuracy of labels assigned to pairs of pixels in segments of two segmentations	↑
GCE	The extent to which one segmentation can be viewed as a refinement of the other	↓
BDE	The average displacement error of boundary pixels between two segmented images	↓
SC	The overlap of regions of the segmentation output, and the regions of the ground truth	↑

have activated 100 pixels for the "vegetation" class and 1000 for the "road" class. Then, we associate each cluster with the class they activated the most, creating an associating dictionary for all of the K classes. Finally, we feedforward the images through the network again while converting the classes with their activations.

Regarding this process, we make a couple of considerations. First, the whole class mapping could be performed with one feedforward operation by storing the network activations. Still, since we use a high value for K, keeping every activation for every image becomes unfeasible. The second is that ideally, the feedforward step should be performed in the training dataset and later applied to the test dataset, but since in unsupervised semantic segmentation real applications, there are no labels, we assume that if a semantic mapping is performed, ground truth labels will be available to be assigned. Also, regarding this second consideration, when the train and test subsets are obtained by splitting the data in a random and stratified way, we can assume that both will have the same, or a very similar distribution, also justifying the mapping being performed in the test set, since it would not have much difference than if performed in the training set.

4.3 Datasets

In this Section we detail the datasets for each experiment, along with a description and a motivation for why we chose each specific dataset. We then show examples of images for every dataset, along with a comprehensive table with several technical details about the datasets.

For all of our experiments regarding the Unsupervised Semantic Segmentation pipeline, we use the Potsdam dataset [ISPRS, 2016]. We chose this remote sensing dataset because it is composed of mostly an urban area, with good resolution images, so it is easier for our network to learn and it is also in a more controlled environment, allowing us to better test our network ability to learn without much variation from the data. This dataset is also widely used in literature, allowing us to compare the results with other methodologies.

As for the automatic scribble generation and weakly supervised experiments, we use the LandCover.ai dataset [Boguszewski et al., 2021] since it is focused mainly on large and spaced rural areas with different landscapes, along with a few smaller urban regions, making it closer to our desired use case, where it will be easier for both humans and our methodology to find large homogeneous areas to draw scribbles. It also proposes a good challenge for the network due to the high variance of some forest areas and the urbar areas, due the difficulty of labeling them.

For the applications, we propose to use two different datasets. The first one is a rural dataset from the region of São Carlos, in Brazil, the same used in Jaimes et al. [2022]. This application evaluates the ability of our model to separate different terrain types, so we chose this dataset because it originally contains 14 classes, where many of these classes are composed of large homogeneous regions of a single type of terrain, therefore being easier to identify the edges that separate different terrains.

And for the second application, we use a dataset from the city of Campinas, in Brazil, to evaluate the model's ability to deal with urban regions, properly separating and segmenting different city areas. In this dataset, the image did not have the R-G-B channels; instead, it had the IR-R-G channels, so it is also an exciting challenge to check whether our model can work in another color space.

An example image of each dataset can be seen in Figure 8, where some of the difficulties of each dataset can be analyzed and compared.

The specifications for each dataset used in this work are present in Table 2, separated by the location of the dataset, the resolution of the images, the land coverage, the geospatial image sizes, the labels present in each dataset, the type of annotation and the geographic features present in the images.

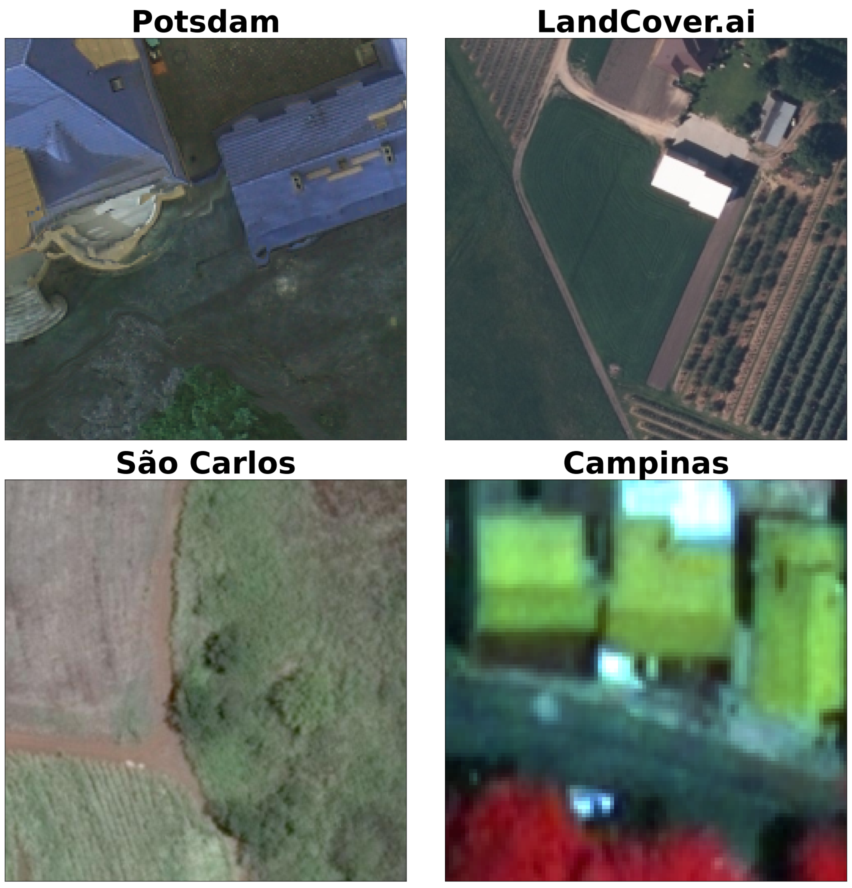


Figure 8 – Examples of images from each dataset used in this work.

Source: produced by the author.

Table 2 – Details for each dataset used in this work.

Dataset	**Potsdam**	**LandCover.ai**	**São Carlos**	**Campinas**
Location	Potsdam - Germany	Poland	São Carlos - Brazil	Campinas - Brazil
Resolution (cm/px)	5	25/50	50	30
Coverage (km²)	∼1.4	216	∼25.6	∼423.57
Image Size (px)	38 images of 6000 x 6000	2 images of 9000 x 9500 and 4200 x 4700	10386 x 9872	69529 x 67688
Labels	roads, buildings, low vegetation, trees, cars, clutter	buildings, woodlands, water, roads, background	river, soil, grass and forest	asphalt, tree, cement, roof 1, roof 2, roof 3, grass, swimming pool, soil
Annotation	semantic masks	semantic masks and weak labels	semantic masks	weak labels
Geographic Features	urban areas	rural areas, rivers, native vegetation, small urban areas	rural areas, rivers, native vegetation	urban areas

Chapter 5

Results

In this chapter, we present the results of every experiment performed. We start with the unsupervised semantic segmentation, then the automatic scribble generation, and finally, the remote sensing applications. For each section, in each experiment, we tested different hypothesis, that will be detailed accordingly.

5.1 Unsupervised Semantic Segmentation

For this pipeline, since the main questions such as "is it possible to group regions with similar features?", or "is it possible to perform semantic segmentation whitout labels?" have already been thoroughly answered, our main hypothesis is: "by analyzing every work in current literature, could we propose a novel pipeline that incorporates the best technologies from different researches, while also proposing our own contributions?". We centered our efforts in the work of Kanezaki [2018], exploring posterior works that derived from it, until the current state-of-art.

Our individual experiments aim to analyze different methodologies in literature by modifying the contents of every block in the original Kanezaki pipeline, to understand in detail how each contributes to the network and what changes in them could lead to more complex semantic information learned. For that purpose, in this section we train several networks from scratch, without any form of transfer learning, so that we can evaluate each of them individually, focused only in our desired tasks. We also show the results of our proposed semantic segmentation pipeline, shown in Figure 2. We start with the Feature Extractor block, the Classifier, the Superpixel Refinement, and finally the Image Preprocessing.

5.1.1 Feature Extractor

Before testing the feature extractor, we must define which architectures were used and other training parameters. The first one was the feature extractor from Kanezaki

[2018], the first work we based ours on. Their feature extractor comprises three blocks; each block includes a Convolutional layer, a ReLU activation layer, and a Batch Normalization layer, except for the last block that does not have a ReLU layer. The constant K is the number of classes chosen, which the authors set to $K = 100$. A simplified version of their pipeline, highlighting the feature extractor, can be seen in Figure 9.

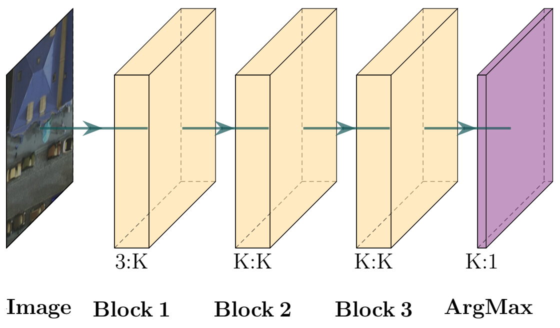


Figure 9 – Simplified pipeline from Kanezaki [2018]. Each block comprises Convolutional, ReLU, and Batch Normalization concatenated layers.

Source: produced by the author.

The second feature extractor tested is derived from Ilyas et al. [2020]. This feature extractor consists of a simple architecture, with an additional squeeze and excite block, similar to what is found in a MobileNet. Figure 10 shows a simplified view of the architecture. Each block comprises a Convolution Layer, a Batch Normalization Layer, and a ReLU layer. As for the Squeeze and Excite block, the activation function for the Linear 1 layer is ReLU, and for the Linear 2 layer is Sigmoid. Note that the block names are repeated twice; the authors use the same blocks twice to extract features and sum the contributions from each of the two forward passes through the blocks.

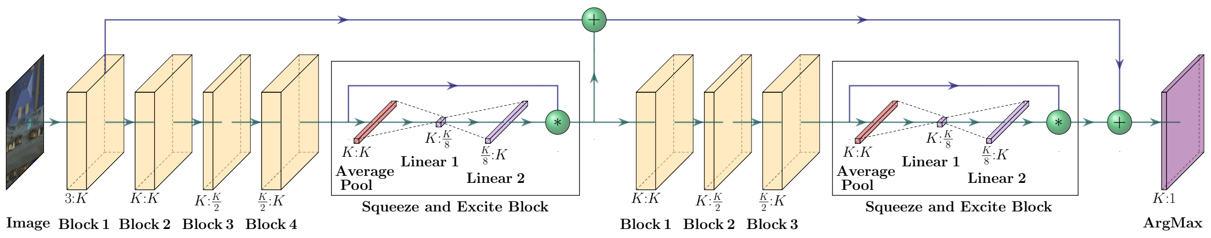


Figure 10 – Simplified pipeline from Ilyas et al. [2020]. Each block comprises Convolutional, Batch Normalization, and ReLU concatenated layers.

Source: produced by the author.

The third architecture is derived from Zhou and Wei [2020]. This architecture is shown in Figure 11. It mainly comprises a few blocks from U-Net, precisely a combination of the first pooling and final upsampling, becoming a simplified version of U-Net, for the first part. Each block comprises a Convolutional layer, a Batch Normalization Layer, and a ReLU layer, except Block 7, which does not have a ReLU activation. The innovative part of the pipeline is the DCS block, which will be discussed next.

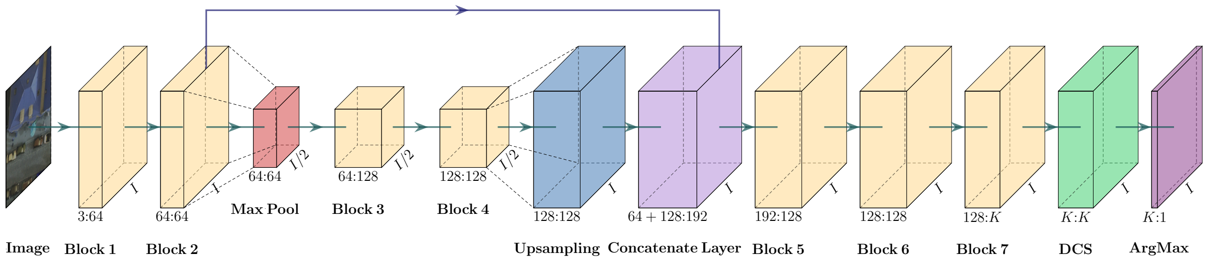


Figure 11 – Simplified pipeline from Zhou and Wei [2020]. Each block comprises Convolutional, ReLU, and Batch Normalization concatenated layers.

Source: produced by the author.

The Deep Subclustering Network block performs clustering of the features before the argmax step. The pipeline is presented in Figure 12. Consider a batch of 1 image, given an input feature block of $K \times I \times I$ features, where K is the given number of classes and $I \times I$ is the dimension of the input image, where width and height are equal here for simplification and consistency with the previous notation, but not required. First, the block goes through a 1×1 convolution, then reshapes to size $Y = I^2 \times K$. The main idea is to associate each pixel in the input image with a cluster among M clusters, then update the cluster centers, reassociate, and repeat these steps several times. This is similar to the K-means algorithm but uses a kernel-based similarity metric. Second, the Associate Rate at step t is calculated by

$$H_{im}^t = \frac{\kappa(Y_i, \Omega_m^{t-1})}{\sum_{j=1}^{M} \kappa(Y_i, \Omega_j^{t-1})}, i \subseteq 1, \dots, I^2, m \subseteq 1, \dots, M \tag{5.1}$$

where Ω_m is the center of the m-th cluster, and k represents a kernel function, which is simply $exp(a^T b)$. In the neural network, this equation is represented as

$$H_t = softmax(Y(\Omega^{t-1})^T)$$

As for the cluster centers Ω, they are updated by

$$\Omega_k^t = \frac{\sum_{i=1}^{I^2} H_{ik}^t Y_i}{\sum_{m=1}^{M} H_{mk}^t} \tag{5.2}$$

These steps are repeated three times for every forward pass in the network. Finally, after the final update, the resulting aggregated features are formulated as

$$\Upsilon = H^t (\Omega^t)^T \tag{5.3}$$

Then, after a ReLU activation layer, another 1×1 convolutional layer, and another ReLU layer, the final output is returned to be passed to the argmax layer.

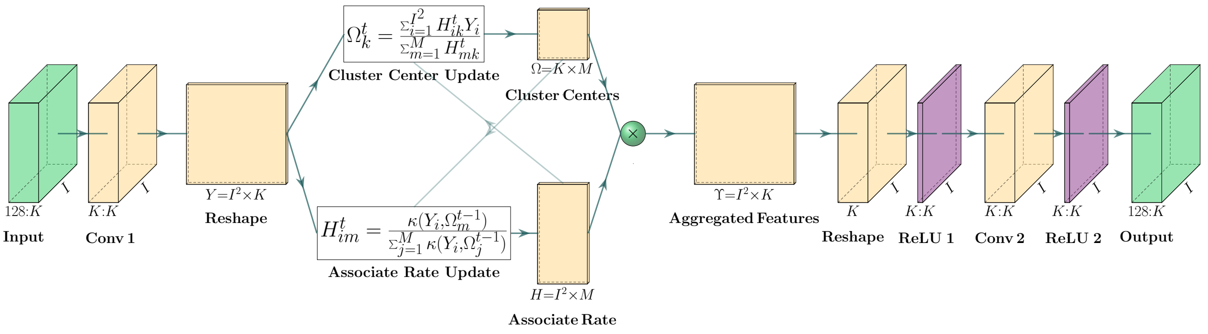


Figure 12 – Deep Clustering Subnetwork block from Zhou and Wei [2020]. Each block comprises Convolutional, ReLU, and Batch Normalization concatenated layers.

Source: produced by the author.

After defining the feature extractors, we proceed to the experiments. We take the following for each of the blocks:

- Preprocessing: the images are normalized between 0 and 1. Since data augmentation is random and we wanted to make the comparisons fair while analyzing the unbiased ability of the feature extractors to learn semantics, we disabled data augmentation for these experiments. However, it is still a part of our pipeline.
- Feature extractor: the three feature extractor blocks described above will be tested individually without any modifications.
- Classifier: a softmax layer with the proposed cross-entropy loss, weighted by the class weights of each batch.
- Class refinement: the output of the unsupervised semantic segmentation from Jaimes et al. [2022] is used to guide the training, and the networks all have the number of clusters set to $K = 100$.

Due to computational resource limitations, we limit each experiment to 10 epochs and the batch size to 10 images. The batch size is chosen due to GPU limitations. As for the number of epochs, it was selected because we empirically observed that the networks mainly converged with this number of epochs, probably because of the high number of images, which will be shown next, leading to a faster generalization.

Moreover, we used the Potsdam dataset for every experiment, as explained in Section 4.3. For the training and testing division, according to literature, most works used the subset of the dataset provided by Ji et al. [2019], where the authors used a total of 8550 images, with 855 images for testing, with complete ground truth, and 7695 images for training, where 4545 also had a complete ground truth and 3150 images had no source of supervision.

Ideally, using these training images with ground truth as a cross-validation set and through our class mapping, we could convert the pixel refinement mask to the actual labels and obtain the accuracy or any other supervised metrics in real-time, at every epoch, allowing curves that could guide the training to be traced. But since our evaluation method is too expensive, which would highly increase the training time, we chose only to use the ground truth for the test set to obtain metrics and to use all the training images without supervision during training, excluding the cross-validation set.

The original Potsdam dataset has six classes: roads, buildings, low vegetation, trees, cars, and clutter. But the authors who created the division of the dataset also proposed a variation where some classes are combined: roads and cars, vegetation and trees, buildings and clutter. This modified dataset is called Potsdam-3, and we also test our algorithms in this reduced version of the dataset.

5.1.1.1 Comparison

To compare the feature extractors, we decided to use an exponential decaying learning rate from $1e^{-4}$ at the beginning of the training until the $1e^{-5}$ at the end so that the network could perform smaller steps toward the gradient as the clustering process reached convergence. A detailed study of the learning rate and information regarding the choice for the number of classes and our loss function can be found in Appendix A.

The metric results for the test dataset can be seen in Table 3, where "FE" stands for "Feature Extractor," and the remaining columns are the abbreviation of each metric, as described in Section 4.1. The "Kanezaki" feature extractor was also abbreviated for "Kan." For every feature extraction, there are two results, one for the complete dataset with all six classes and another where the name of the feature extractor has a "3", meaning that it is the variation of the dataset with only three classes, both separated by a thick horizontal line. In this table, the IoU and wIoU metrics are the mean over all the classes, and the segmentation quantitative metrics (VoI, PRI, GCE, BDE, and SC) are all the mean of the metric value for every image in the test dataset. We also separate the types of metrics with a thicker vertical line between $F1$ and VoI, where the left half represents the supervised metrics, and the right half represents the segmentation metrics.

As the results show, the DIC feature extractor obtained the best supervised metrics for both datasets, as depicted on the left side of the table. Checking the segmentation metrics, we can see that VoI, GCE, and SC are better for SEEK, while PRI and BDE are better for DIC.

In a more in-depth analysis of the segmentation metrics only (right half of the

Table 3 – Metrics obtained for different feature extractors.

FE	**Acc↑**	**IoU↑**	**wIoU↑**	**F1↑**	**VoI↓**	**PRI↑**	**GCE↓**	**BDE↓**	**SC↑**
SEEK	58.6	24.6	38.6	33.0	**1.58**	0.678	**0.217**	12.6	**0.567**
DIC	**59.4**	**32.4**	**43.2**	**45.7**	1.84	**0.698**	0.293	**11.1**	0.548
Kan	58.4	26.8	40.0	37.3	1.83	0.686	0.293	11.2	0.535
SEEK3	72.8	56.4	57.4	71.6	**1.27**	0.733	**0.202**	14.5	**0.688**
DIC3	**74.1**	**58.4**	**59.4**	**73.2**	**1.27**	**0.740**	0.203	**13.8**	0.679
Kan3	72.7	56.4	57.2	71.7	1.40	0.717	0.225	14.8	0.667

table), the metrics highlighted for SEEK mostly show that the segmentations achieved are closer to the ground truth in their shape. As for DIC, the highlighted metrics indicate that the segmentation is closer to the ground truth in content.

Given these results, we chose to use the DIC feature extractor since it obtained the most consistent supervised and competitive segmentation metrics compared to the others.

5.1.1.2 Second Classifier

With the previous experiment, we had already decided on our feature extractor, but we decided to test another idea that could potentially improve our methodology. Many different works use two classifiers during learning, either to maximize their mutual information [Ji et al., 2019, Harb and Knöbelreiter, 2021] or as a form of data augmentation and regularization, to learn multiple representations of an image at the same time [Cho et al., 2021]. Either way, learning with two classifiers simultaneously can be beneficial and create the possibility of exploiting this information as a loss function since both would theoretically be learning to segment the same images.

With this idea in mind, we modified our feature extractors to allow two classifiers in parallel. In practice, our solution is close to Ji et al. [2019], except that he uses a different number of classes for each classifier while we set the same number for both.

To add the second classifier, we modify the DIC feature extractor, shown in Figure 11, by adding a replicate of Block 7, the DSC block, and the ArgMax layer, working in parallel with the original blocks. The output of Block 6 goes to both the original blocks and the replicate blocks simultaneously, and accordingly, two classifications can be performed simultaneously.

We first show in Figure 13 the loss curves for both classifiers for the training, where "C1" stands for "Classifier 1" and "C2" for "Classifier 2". We used the optimal learning rate and loss function defined in the previous experiments.

As expected, every loss function achieves a similar loss and shape at the end of the training, even though they start differently due to the weights initialization, showing

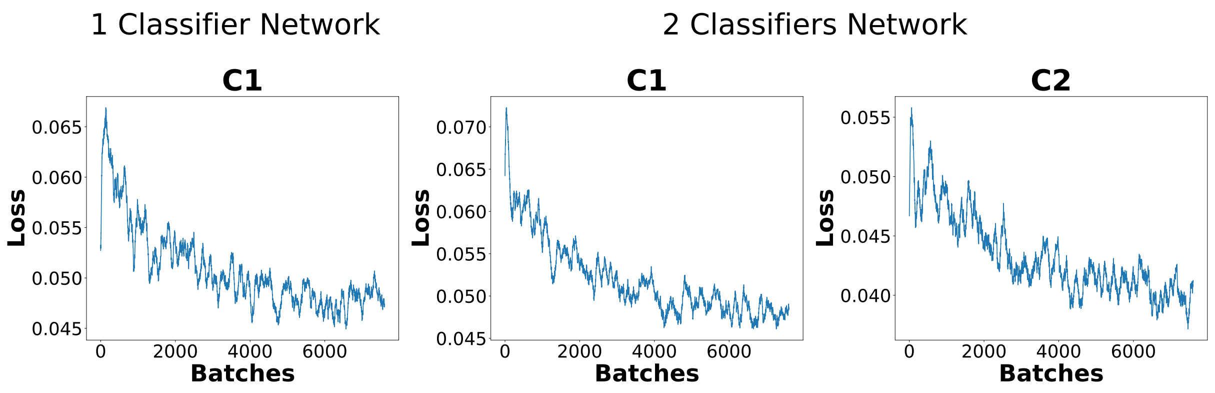


Figure 13 – Losses for the multiple classifiers experiment.

Source: produced by the author.

that all classifiers are learning correctly and indicating that they are learning the same segmentation, which can only be proved when comparing the actual segmentations. Then we show the metrics results in Table 4, where "NC" stands for "Number of Classifiers." In the Feature Extractor column, the "_1" indicates the first classifier, and equally, "_2" is the second classifier.

Table 4 – Metrics for the multiple classifiers experiment.

NC	FE	Acc↑	IoU↑	wIoU↑	F1↑	VoI↓	PRI↑	GCE↓	BDE↓	SC↑
1C	DIC_1	59.4	32.4	43.2	**45.7**	1.84	0.698	0.293	11.1	**0.548**
2C	DIC_1	**60.9**	32.5	44.1	45.5	**1.75**	**0.708**	**0.281**	11.4	0.541
	DIC_2	**60.9**	**32.6**	**44.4**	45.5	1.79	0.706	0.286	**11.0**	0.547
1C	DIC3_1	74.1	58.4	59.4	73.2	1.27	0.740	0.203	**13.8**	0.679
2C	DIC3_1	**75.7**	**60.3**	**61.2**	**74.8**	**1.19**	**0.754**	**0.190**	14.0	**0.693**
	DIC3_2	**75.7**	**60.3**	**61.2**	**74.8**	1.23	0.748	0.196	14.6	**0.693**

Overall, in almost every metric, the experiment with two classifiers obtained a better result, having increases of about 1.5% for the accuracies and similar increases for every other supervised metric. The only metrics with a better outcome for the one classifier experiment were the F1-score and SC for the six classes dataset and BDE for the three classes dataset. However, the results are still close, with none surpassing differences of 0.2%.

Additionally, if we analyze the results of both classifiers of the second experiment, they are almost equal in every metric, with slight variations. This confirms that they are both learning the same segmentation and is coherent with what we expected. As for how they are learning this segmentation, this could vary for each classifier. Since it is an unsupervised problem, the cluster might differ from each classifier in practice since the metrics only evaluate the results after the softmax activation. This can only be assessed when checking the actual segmentations, which will be performed later.

5.1.2 Pixel Classifier

After defining the optimal feature extractor for our network, along with several hyperparameters, we test the addition of other losses to our proposed weighted cross-entropy loss (L_{wce}) to check if it is possible to further enhance the semantic extraction with additional constraints to the training.

The first loss we propose to use is the spatial continuity loss (L1 loss), presented by Kim et al. [2020]. Since clustering primarily aims to group pixels, creating homogeneous regions in the segmentation mask, the network output should benefit from being spatially continuous, forcing the cluster labels to be similar for neighbor pixels. For that, the authors use the L1-norm of the horizontal and vertical differences of the response map (r) or the logits of the network classification before the softmax activation. The loss is given by

$$L_{con} = \frac{1}{K \cdot (H-1) \cdot (W-1)} \sum_{n=1}^{K} \sum_{i=1}^{H-1} \sum_{j=1}^{W-1} \|r_{n,i+1,j} - r_{n,i,j}\|_1 + \|r_{n,i,j+1} - r_{n,i,j}\|_1 \tag{5.4}$$

where the $\|\cdot\|$ is the norm-1, or simply the absolute value of the difference, $r_{n,i,j}$ is position (i, j) of the n-th response map, K is the number of clusters, or classes, H is the image height and W is the image width.

The second loss we implemented is the Mumford-Shah loss (MS loss), proposed by Kim and Ye [2020]. Their loss is based on the Mumford-Shah functional and tries to minimize the energy variation within the pixels in each cluster. In practice, for each of the K classes, a centroid is calculated using its probability map applied to the color image. Then each pixel is subtracted from the color of its centroid, and this difference is minimized. Along with this minimization, the authors also add a constraint to enhance the bias field continuity, that is, the L1-norm of the horizontal and vertical differences of the activations of the response map. This is almost the same as the spatial continuity loss from Kim et al. [2020], except that the L1-norm is taken after the softmax activation output instead of the response map.

Given a pixel (r), an input image (x) and the softmax output (y), the loss is calculated by

$$L_{MScnn}(\Theta, x) = \sum_{n=1}^{K} \sum_{r=1}^{N} |x(r_i) - c_n|^2 y_n(r_i) + |\nabla y_n(r_i)| \tag{5.5}$$

where N is the total number of pixels, Θ represents the network parameters, $|\nabla y_n(r)|$ is the L1-norm of the softmax output, and c_n is the centroid calculated using the input image and softmax activations as membership probabilities for each class

$$c_n = \frac{\sum_{r=1}^{N} x(r_i) y_n(r_i)}{\sum_{r=1}^{N} y_n(r_i)}$$

The resulting loss curves for the network trained with each of these losses are shown in Figure 14. Note that the total loss of our network is composed of either the sum of our weighted cross-entropy loss with the spatial continuity loss.

$$loss_1 = L_{wce} + L_{con}$$

or by the sum of our loss with the Mumford-Shah loss

$$loss_2 = L_{wce} + L_{MScnn}$$

In the following Figures, we evaluate how adding another loss affected the shape of the curve.

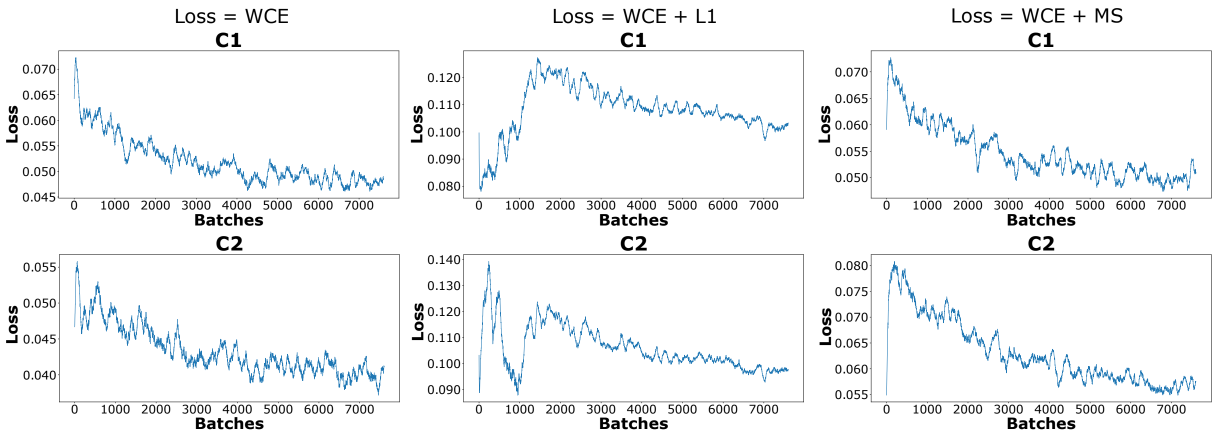


Figure 14 – Comparison between different losses used to train the network.

Source: produced by the author.

When adding the L1 loss, the curves indicate that the network had more difficulty learning since it converges slower. As for the MS loss, the curve seems to converge similarly, with only the loss value being higher because both losses are being summed up. Next, we show the metrics in Table 5, where we abbreviate the classifier (C) as being either 1 (C1) or 2 (C2). Also, we did not specify the dataset here for lack of space, but the upper half shows the metrics for the dataset with six classes and the bottom half for the dataset with three classes.

We can see that with the L1 loss, the segmentations improved a bit since the Acc, VoI, PRI, and GCE were generally better in the six classes. By enforcing homogeneous regions, we can assume that larger classes got a better representation, but at the cost of sacrificing smaller classes. This is similar to the dataset with three classes, except the

Table 5 – Metrics for the different losses experiment. The first half shows the results for the Potsdam dataset, and the second half for the Potsdam-3 dataset.

Loss	C	Acc↑	IoU↑	wIoU↑	F1↑	VoI↓	PRI↑	GCE↓	BDE↓	SC↑
WCE	C1	60.9	32.5	44.1	**45.5**	1.75	0.708	0.281	11.4	0.541
	C2	60.9	**32.6**	**44.4**	**45.5**	1.79	0.706	**0.286**	**11.0**	**0.547**
WCE+L1	C1	**61.7**	29.5	43.7	40.0	**1.62**	**0.715**	0.257	11.4	0.534
	C2	61.4	29.3	43.5	39.8	**1.62**	0.714	0.257	11.7	0.539
WCE+MS	C1	60.5	29.1	42.5	40.1	1.74	0.703	0.284	11.2	0.535
	C2	60.8	29.3	42.8	40.2	1.72	0.706	**0.286**	11.2	0.528
WCE	C1	**75.7**	**60.3**	**61.2**	**74.8**	1.19	**0.754**	0.190	**14.0**	0.693
	C2	**75.7**	**60.3**	**61.2**	**74.8**	1.23	0.748	0.196	14.6	0.693
WCE+L1	C1	74.1	58.4	59.2	73.4	**1.17**	0.752	**0.185**	14.8	0.703
	C2	74.1	58.2	59.0	73.2	**1.17**	**0.754**	**0.185**	15.1	**0.704**
WCE+MS	C1	73.8	57.6	58.5	72.7	1.27	0.741	0.201	14.5	0.687
	C2	74.2	58.1	59.1	72.9	1.23	0.748	0.194	**14.0**	0.683

Acc. As for the MS metrics, almost no metric improved, indicating that this loss did not improve the model. With these results, we opt to stay with WCE loss only.

5.1.3 Class Refinement

Our final experiment with our network regards the class refinement step. All previous studies in the literature have adopted a superpixel segmentation to enforce the pixels within each superpixel to share a common class. We proposed using our class refinement, with the segmentation provided by the external algorithm of Jaimes et al. [2022], where it uses superpixels to segment the image but applies several pre and post-processes to obtain a more consistent segmentation of the image instead of the raw superpixels.

In this experiment, we compare our network's training using superpixels and our proposed mask. The losses for each training are shown in Figure 15, where "GT" indicates which type of Ground Truth labels was used to train the network.

We can see by the results that the loss is much smoother for the superpixel experiment, achieving convergence with less noise. This can be explained because when using an external algorithm, a previous clustering is performed, so the labels enforce classes that can have significant differences in size. As for the second experiment, since multiple superpixels are not enforced to be in the same class, the network can choose the class enforcement by itself, resulting in a more negligible difference in the class sizes. This is directly reflected in the noise in each loss function, much of which can be attributed to our weighted cross-entropy loss. We then compare the metrics for each experiment in Table 6, where "Seg" stands for Segmentation method, "EX" for "External algorithm," and "SP" for "Superpixels."

Overall, only a few segmentation metrics had a better value for the External Algorithm method, and even those were close to the Superpixels results. These results show

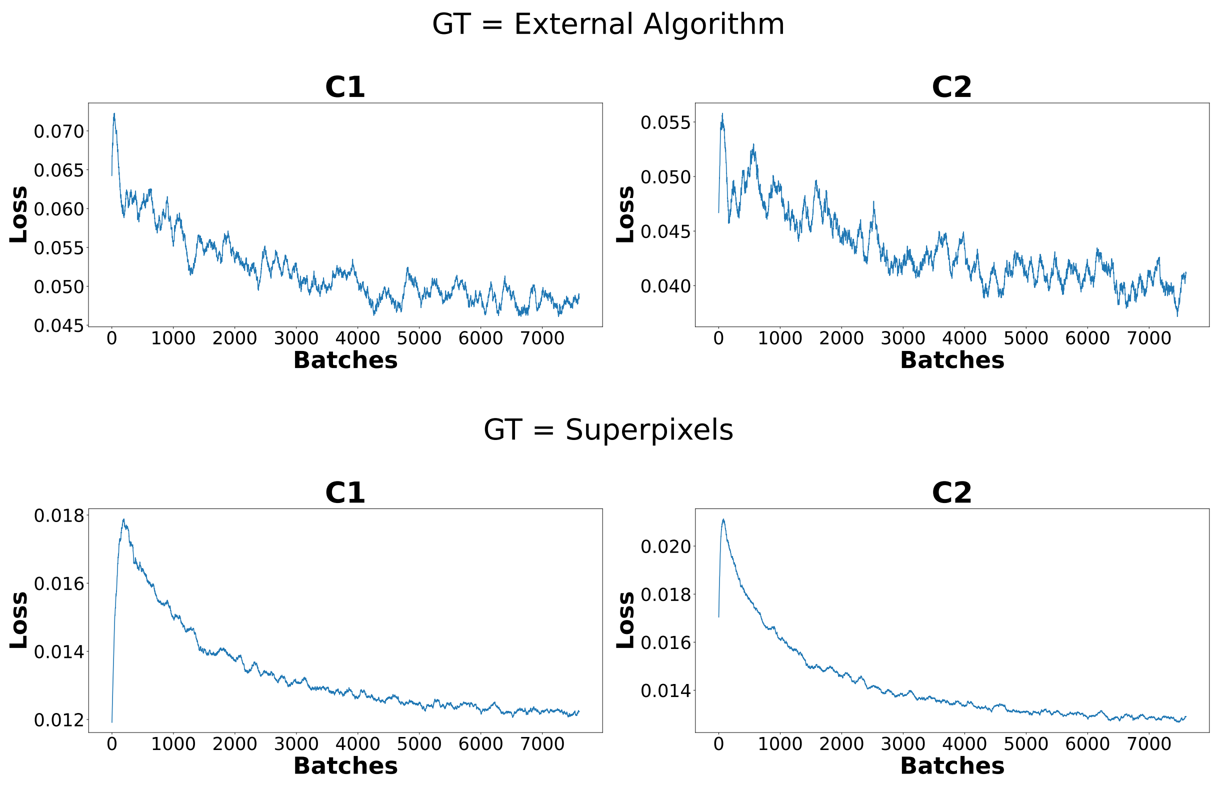


Figure 15 – Losses for different types of ground truth labels.

Source: produced by the author.

Table 6 – Metrics for the class number experiment.

Seg	C	Acc↑	IoU↑	wIoU↑	F1↑	VoI↓	PRI↑	GCE↓	BDE↓	SC↑
EX	C1	60.9	32.5	44.1	45.5	1.75	**0.708**	0.281	11.4	0.541
	C2	60.9	32.6	44.4	45.5	1.79	0.706	0.286	11.0	0.547
SP	C1	62.6	32.3	45.2	44.7	**1.74**	0.705	0.281	10.8	0.551
	C2	**62.9**	**33.1**	**45.8**	**45.8**	1.78	0.706	**0.278**	**10.5**	**0.566**
EX	C1	75.7	60.3	61.2	74.8	**1.19**	0.754	**0.190**	**14.0**	0.693
	C2	75.7	60.3	61.2	74.8	1.23	0.748	0.196	14.6	0.693
SP	C1	75.9	60.4	61.3	74.9	1.26	**0.747**	0.199	14.1	0.690
	C2	**76.3**	**61.0**	**61.8**	**75.4**	1.22	0.750	**0.190**	14.1	**0.696**

that even though providing a pre-grouping of the classes could help to lead the network training, letting the network learn these groupings by itself is even better, generating a final semantic segmentation closer to supervised ground truths. Therefore, we abandon the External Algorithm idea for now and stay with the broadly used superpixel segmentation.

5.1.4 Image Preprocessing

After defining the final specification of the network, the only block missing is the Image Preprocessing. We leave the preprocessing experiment for last because we needed the complete architecture.

To test whether the network needs a preprocessing block or not, we perform an additional experiment were we train 4 different networks for 100 epochs instead of 10, so that we may see the effects of the data augmentation. In each experiment, we train with a different configuration of data augmentations:

- 1 - No data augmentation.
- 2 - Only geometric augmentations (for images and labels).
- 3 - Only color augmentations (for images only).
- 4 - All data augmentations.

This way, we expect to be able to observe which type of data augmentation further impacts our model. The results for this experiment can be seen in Table 7.

Table 7 – Experiments with different types of data augmentation.

Exp	**Acc↑**	**IoU↑**	**wIoU↑**	**F1↑**	**VoI↓**	**PRI↑**	**GCE↓**	**BDE↓**	**SC↑**
1	**63.5**	**37.0**	**46.3**	51.6	**1.75**	**0.710**	**0.278**	**10.4**	**0.562**
2	62.1	36.7	45.2	**51.8**	1.80	0.701	0.283	11.0	0.547
3	62.7	33.7	45.5	46.7	**1.75**	0.703	0.280	10.9	0.559
4	60.8	29.2	43.0	40.1	**1.75**	0.707	0.282	11.3	0.545
1	**76.5**	**61.4**	**62.2**	**75.8**	**1.23**	0.747	**0.193**	**14.1**	**0.702**
2	76.1	60.7	61.5	75.3	1.27	0.743	0.200	14.8	0.696
3	76.3	61.2	62.0	75.6	**1.23**	**0.748**	0.194	**14.1**	0.695
4	75.5	59.7	60.5	74.4	1.25	0.744	0.198	15.2	0.686

As seen, the experiment without data augmentations had the best results. Therefore, we may conclude that in this specific case of unsupervised semantic segmentation, adding data augmentation might have introduced more noise to the model, rather than generalizing. Given this result, we choose to exclude data augmentation from our final architecture.

5.1.5 Final architecture

After analyzing every building block and every step, a summary of our final network is presented:

- Preprocessing
  - Image Normalization.
- Feature Extractor
  - DIC feature extractor [Zhou and Wei, 2020] modified to accommodate two classifiers instead of one.

- Classifier
  - Weighted cross-entropy loss, followed by argmax operation.
- Class Refinement
  - Superpixels ground truth.
- Training
  - Exponential decaying learning rate from $1e^{-4}$ to $1e^{-5}$.

Additionally, in Table 8, we compare our metrics with other reported results in the literature to measure our architecture improvements concerning the current state of the art. The columns are "Algorithm" for different algorithms in the literature, "Type" for the type of algorithm, which can be either a Clustering algorithm (Clust), a CNN-based algorithm (such as ours), and the newer Visual Transformers-based approaches (ViT). We only compare the Accuracy (Acc) and IoU for both Potsdam (P) and Potsdam-3 (P3) datasets since these are the results provided in these works.

Table 8 – Comparison of our methodology with other algorithms in the literature.

Algorithm	**Type**	**P-Acc**	**P-IoU**	**P3-Acc**	**P3-IoU**
SIFT*† [Ji et al., 2019]	Clust	28.5	-	38.2	-
DeepCluster*† [Ji et al., 2019]	Clust	29.2	-	41.7	-
K-means† [Ji et al., 2019]	Clust	35.3	-	45.7	-
Doersch*† [Ji et al., 2019]	Clust	37.2	-	49.6	-
Isola*† [Ji et al., 2019]	Clust	**44.9**	-	**63.9**	-
Random CNN [Ji et al., 2019]	CNN	28.3	-	38.2	-
IIC [Ji et al., 2019]	CNN	45.4	-	65.1	-
AC [Ouali et al., 2020]	CNN	49.3	-	66.5	-
InMARS [Mirsadeghi et al., 2021]	CNN	47.3	-	70.1	-
InfoSeg [Harb and Knöbelreiter, 2021]	CNN	57.3	-	71.6	-
SGSeg [Eliasof et al., 2022]	CNN	57.7	-	71.8	-
SAN [Zhang et al., 2022]	CNN	60.5	**-**	-	-
Ours	CNN	**62.9**	**33.1**	**76.3**	**61.0**
SegSort† [Ke et al., 2022]	ViT	59.0	35.0	-	-
HSG [Ke et al., 2022]	ViT	**67.4**	**43.8**	-	-
DINO† [Seong et al., 2023]	ViT	-	-	53.0	-
STEGO [Hamilton et al., 2022]	ViT	-	-	77.0	-
HP [Seong et al., 2023]	ViT	-	-	**82.4**	-

* Clustering of features, the method is not initially designed for image segmentation.
† The cited paper is where the image segmentation experiment was performed and the metrics extracted, but this is not the original author of the algorithm.

The comparison shows that we obtained the best result from the CNN algorithms yet. Even though our methodology performs poorer than the ViT algorithms, which are the current state-of-art, our results are competitive even with them, indicating that we were able to extract the most of the performance for every step of our network.

Additionally, in Figure 16, we showcase some segmentation results obtained by our network, compared with the ground truth, for the Potsdam dataset with six classes. Likewise, we show the same images in Figure 17, but for the Potsdam-3 dataset. The label association using the original dataset colors is:

- Roads: White
- Buildings: Blue
- Low Vegetation: Cyan
- Trees: Green
- Cars: Yellow
- Clutter: Red

In the segmentations for the dataset with six classes, we can see that the network could segment many different shapes properly. The car segmentation is confused with the building class; in another example, the clutter class was confused with the building class. Both problems are a direct problem of the class mapping strategy since the objects were well segmented. Moreover, we can see that the dataset itself has many issues. Buildings and clutter are very similar, and the trees do not have well-defined leaves, so many of them allow you to see the ground beneath, leading to many classification errors in this class. As for the dataset with three classes, many confusions are mitigated, except for the car's class, which is still unrecognizable.

5.2 Automatic Scribble Generation

In this section, we explore the possbility of automatically generate scribbles in images. For that, we have two main hypoteshis, the first related to our unsupervised pipeline for scribble generation: "is it possible to group the homogeneous regions of images and automatically draw scribbles, in a way similar to a human?". And our second hypothesis regards our semi-supervised pipeline for classifying scribbles: "is it possible to train a classifier using features extracted from a smaller set of scribbles, and use it to further classify a larger set of scribbles?". Each pipeline created to answer these questions will be further tested and analyzed.

For the remaining of the section we show the results of the scribble generation proposed methodology. We start with the results for the scribble classifier, followed by the

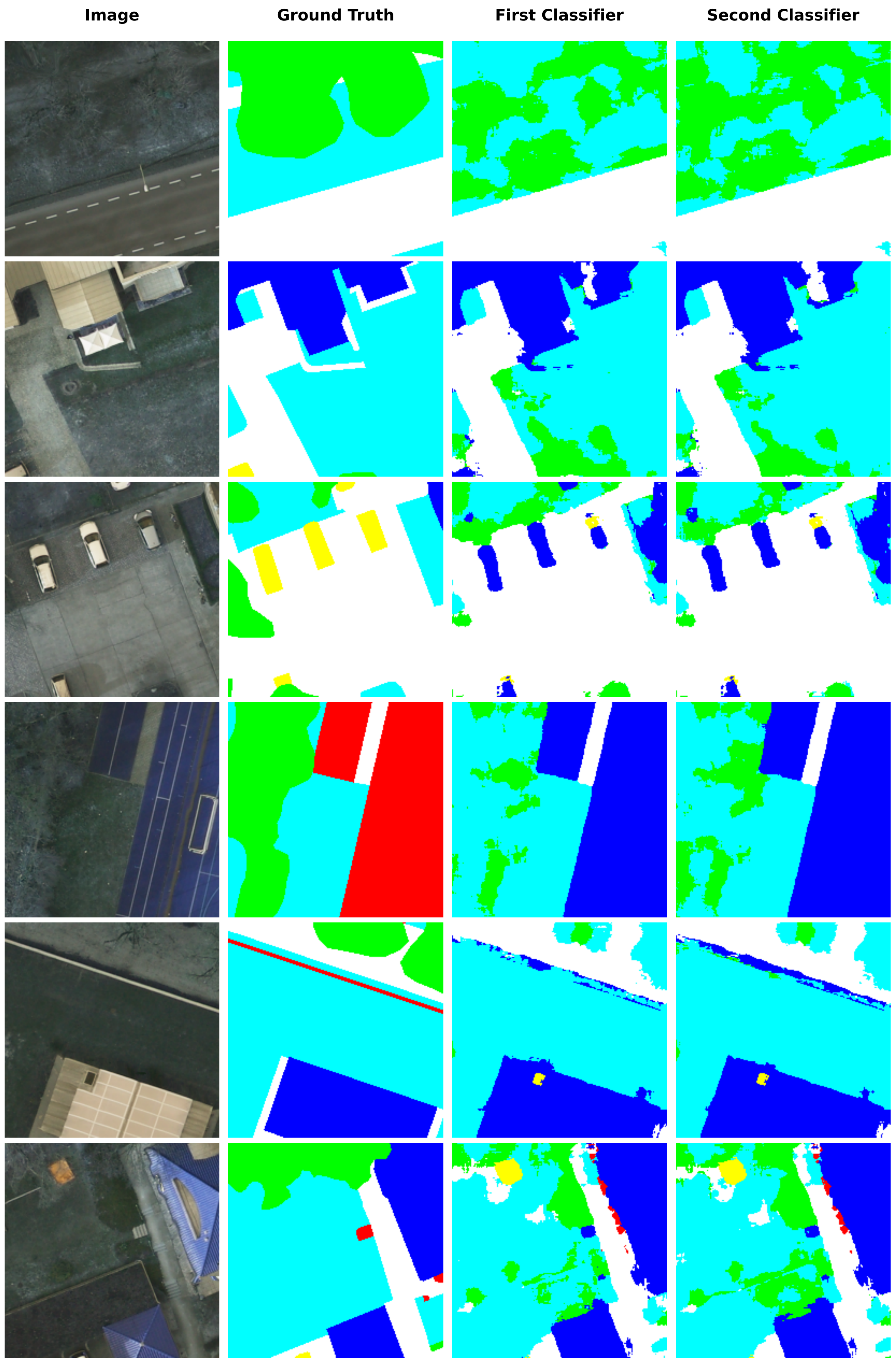


Figure 16 – Examples of semantic segmentations for the Potsdam dataset.

Source: produced by the author.

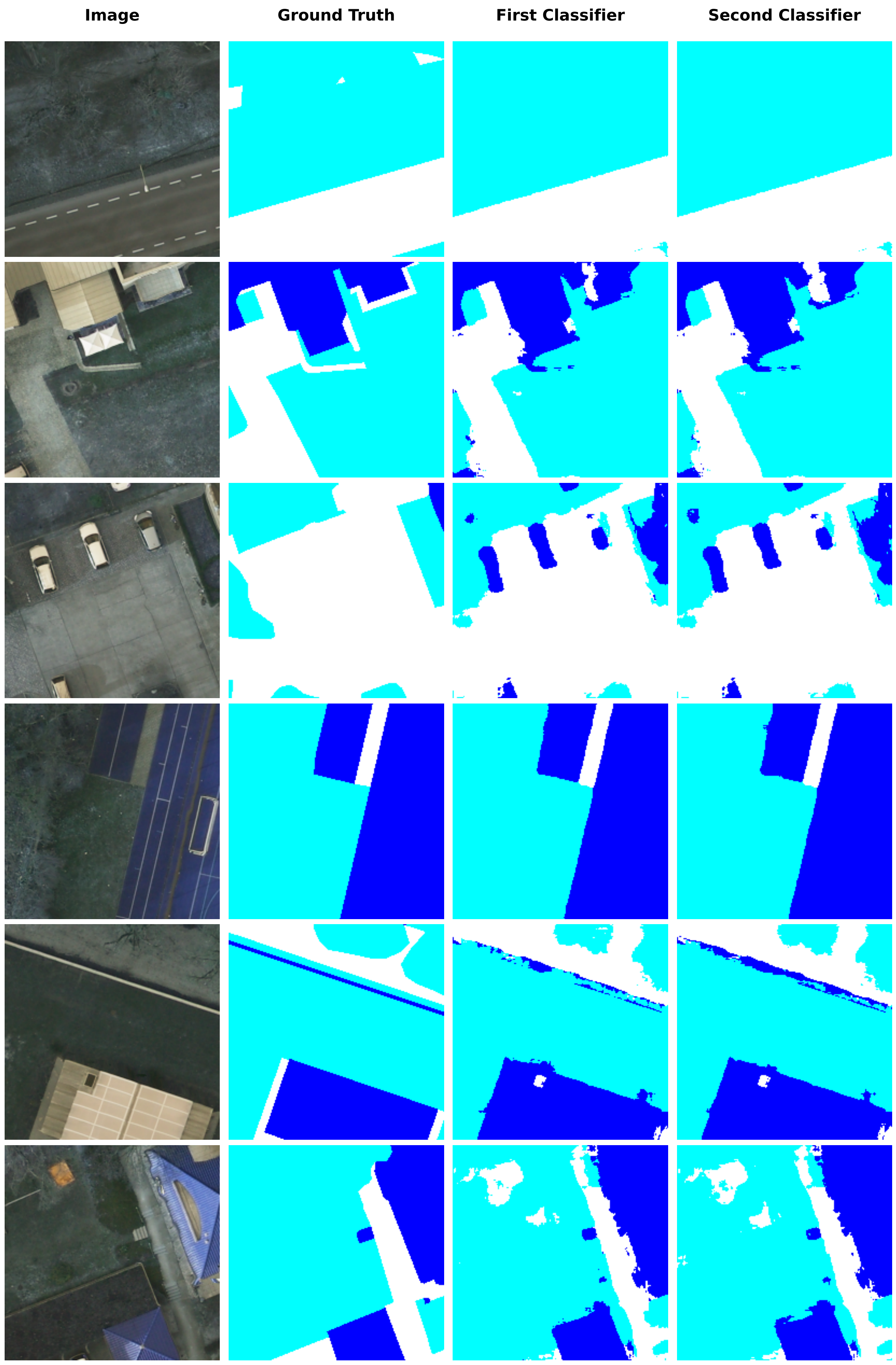


Figure 17 – Examples of semantic segmentations for the Potsdam-3 dataset.

Source: produced by the author.

results of our network trained with the supervision of different types of scribbles. Since the scribble generation is not a quantitative process, we show examples of the automatically generated scribbles and the other experiments' results.

For the dataset, as described in Section 4.3, we used the LandCover.ai dataset [Boguszewski et al., 2021]. A subset of 780 images with a size of 512 × 512 pixels was used for all experiments, which comprehends areas of different characteristics from Poland. This subset was re-annotated using scribbles and later used to train the model. Those images were selected based on their saturation, brightness, and resolution similarity since these conditions represent the entire dataset well. Similarly, 276 images were chosen to compose the test set. The pixel proportion in which the classes appear in the training set is 0.56% for buildings, 61.17% for woodland, 5.07% for water, 1.89% for roads, and 31.31% for background. This is a very unbalanced dataset, providing an additional challenge.

5.2.1 Scribble Classifier

In the pipeline proposed in Figure 6, we suggest annotating the scribbles of a small set of images and training an SVM to automatically classify a large number of automatically generated scribbles. We start by testing if it is possible to classify scribbles based on their extracted features. When automatically generating the scribbles, they are all assigned a generic class. With our provided graphical tool, drawing, editing, removing, and changing the class of already drawn scribbles is possible. This way, the user can auto-generate, label and fix the worst ones.

To experiment with the scribble classifier, we explored two levels of human intervention in the annotation. The first is when an expert draws all scribbles, and the second is when the scribbles are generated automatically. In both cases, an SVM classifier is trained using the mean features (step 1.4) of every superpixel crossed by a scribble. The scribbles of 10% of the dataset images are used, and the remaining 90% are classified, as illustrated in pipelines' steps 2.2 and 2.3. We then set four experiments to evaluate the process of scribble classification.

- Experiment 1 - train the SVM using 10% of the images with human-made scribbles and classify the remaining 90% with automatically generated scribbles.
- Experiment 2 - same as 1 with the building detection module.
- Experiment 3 - train the SVM using 10% of the images with automatically generated scribbles and classify the remaining 90% also with automatically generated scribbles.
- Experiment 4 - same as 3 with the building detection module.

Table 9 shows Precision ("Prec") and Recall ("Rec") achieved by the SVM classifier for each class, with the ground truth, that is, the actual class of the scribbles, provided by an expert. The abbreviations are "Exp" for "Experiment," "Back" for Background, "Build" for Building, and "Wood" for Woodland.

Table 9 – Metrics for the scribble classifier.

Exp	**Metric**	**Back**	**Build**	**Wood**	**Water**	**Road**	**Mean**
1	Prec	85.52	85.45	93.83	50.00	87.34	80.43
	Rec	78.61	90.82	87.49	97.50	66.35	84.15
2	Prec	87.30	73.29	90.88	50.64	81.01	76.62
	Rec	73.29	90.48	87.69	96.60	59.92	81.60
3	Prec	69.89	100.00*	96.62	68.18	65.79	80.10
	Rec	92.14	100.00*	77.95	85.71	32.05	77.57
4	Prec	81.26	73.89	92.25	66.90	64.63	75.79
	Rec	83.94	92.02	88.73	86.61	38.41	77.94

* There were only 3 examples in this class

Generally speaking, the results of experiments 3 and 4 were close to experiments 1 and 2, with a slight worsening in Road class recognition. However, this result shows that the proposed method for automatic generation and classification of scribbles can mimic human-made scribbles well.

The results show that, in general, the best precision and recall were obtained by experiment 1, with human annotations without building detection implemented. Comparing experiments 1 and 2 with human-annotated labels, the performance worsened, even for the building class. Compared with the fully automatic experiments 3 and 4, results generally improved, except for the building class. Still, in experiment 3, the building class only had 3 scribbles, which explains the 100% precision and recall and thus is not reliable. This can be explained because both were generated using similar locations in the feature space. In contrast, the human labels were generated with a different perception, causing the classifier not to generalize well.

When comparing the best human experiments with the best automatic experiments, performance is similar for the Background and Woodland. For the Building and Road, the performance is better for the human labels because these classes involve small and narrow objects, making it more challenging to generate their corresponding scribbles automatically. For the same reason, the worst overall class is "road" in experiments 3 and 4. Despite this, we believe this problem could be solved by manually fixing these scribbles, either by removing the bad ones and redrawing them or just drawing additional ones.

For the Water class, the recall dropped, but the precision increased, meaning that the features extracted by the automatic scribbles in the training set are closer to the ones in the test set, whereas the features from the human scribbles do not represent well the test set.

At last, the building detection module seems to hinder the performance of the scribbles classifier, as can be seen when comparing experiments 1 and 2 or 3 and 4. Some examples of scribbles generated by our methodology can be seen in Figure 18, along with the respective ground truth labels. The color-label association for this dataset is:

- Buildings: Orange
- Woodlands: Green
- Water: Blue
- Roads: Yellow
- Background: Red

5.2.2 Semantic Segmentation

In this experiment, we evaluate both the effect of automatic-generated scribbles in DSSNs training and the usage of weak labels in our proposed optimal DSSN architecture. The same Experiments 1 to 4 from the Scribble Classifier evaluation, presented in Section 5.2.1, are used here. As baselines, DSSNs were trained with the complete supervision of the ground-truth semantic masks, with semantic masks built from a set of 100% human-annotated scribbles, and an utterly unsupervised pipeline with masks generated from superpixels, as proposed for our network in Section 5.1.

In practice, since our network solves a clustering task, and therefore we do not have means of associating the clusters with the actual labels, we adapted the network to perform with the scribbles the same as in superpixels: group every pixel inside scribbles of the same class to be the same cluster. With this approach, we do not have an actual supervised loss, but we can still evaluate the quality of the scribbles and use them as supervision for our network.

The dataset for these experiments contains 780 images for training and 276 images for testing. Due to our computational limitations, the images in this dataset have 256×256 pixels, so we used a batch size of 8 images. In a real case scenario, we would combine both the unsupervised and weakly supervised losses. Still, since we want to compare the generated scribbles' ability to provide proper supervision for an algorithm, we use the scribbles as the only source of supervision to train the network in the proposed experiments. As for the number of epochs, since this dataset has a low number of images and weak training with only weak labels requires a more extensive training time to achieve convergence, we train our models for 100 epochs to provide a sufficiently high number of epochs to obtain a good generalization.

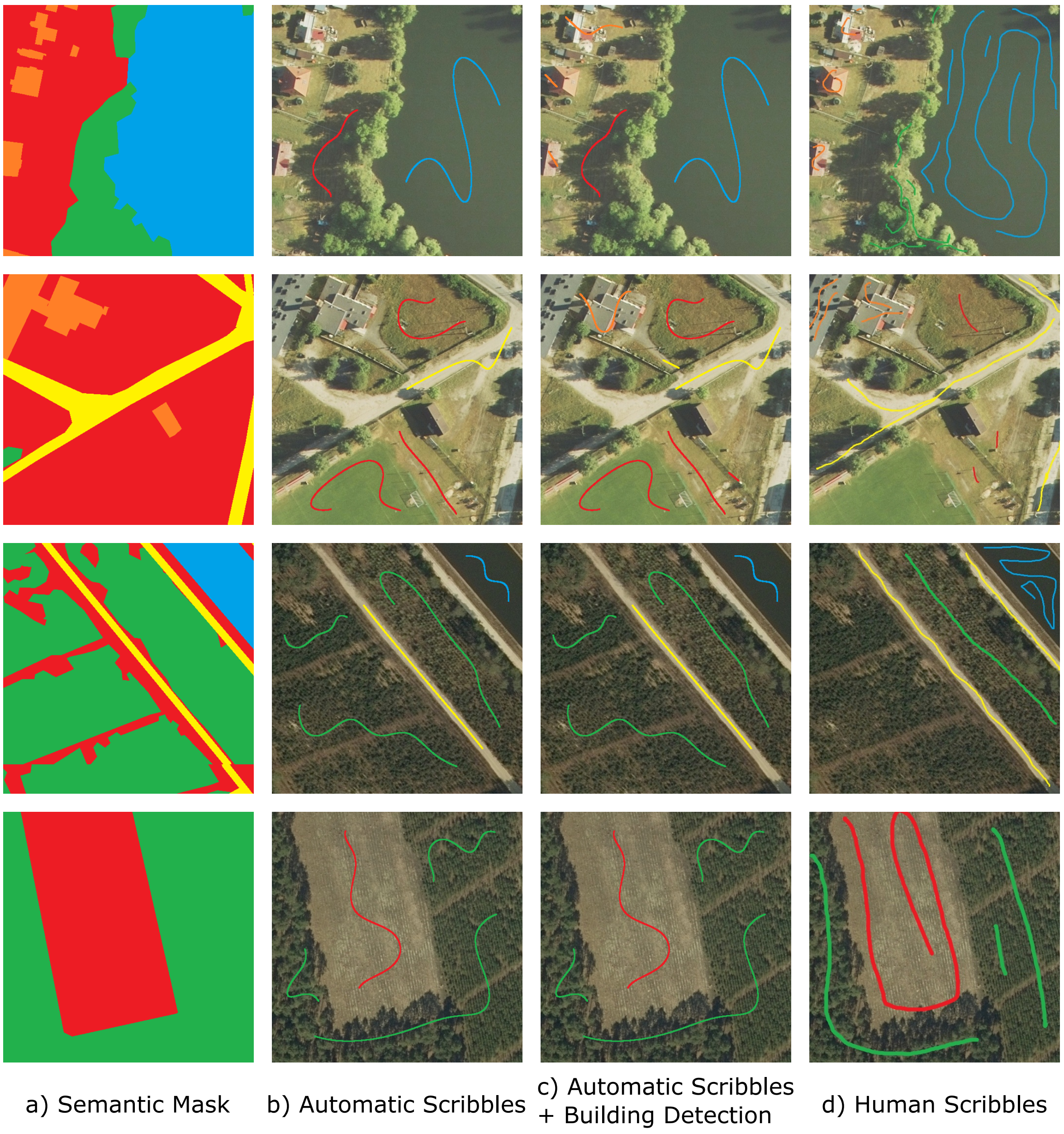


Figure 18 – Different types of scribbles comparison.

Source: produced by the author.

The metrics for the test set are shown in Table 10, where the experiments are the same as the previous section, "HAS" stands for Human Annotated Scribbles, "Sup" for Supervised, and "Unsup" for Unsupervised.

Table 10 – Metrics for the class number experiment.

Exp	C	Acc↑	IoU↑	wIoU↑	F1↑	VoI↓	PRI↑	GCE↓	BDE↓	SC↑
1	C1	81.9	34.3	68.8	42.6	0.85	0.775	0.116	30.4	0.754
	C2	82.0	34.2	69.0	42.6	0.87	0.771	0.121	30.8	**0.773**
2	C1	83.2	34.8	70.8	42.9	**0.81**	0.785	**0.109**	28.5	0.761
	C2	**83.7**	35.5	71.6	43.5	0.82	0.786	0.112	30.7	0.767
3	C1	82.7	33.9	71.0	41.7	0.89	0.782	0.123	30.6	0.762
	C2	83.0	35.4	70.8	43.6	0.83	0.786	0.114	30.3	0.734
4	C1	83.1	36.5	71.1	44.6	0.86	0.782	0.121	28.3	0.720
	C2	83.6	**36.9**	**72.0**	**44.9**	0.85	**0.789**	0.120	**28.1**	0.725
HAS	C1	87.2	37.6	77.7	44.6	0.74	0.832	0.105	25.1	0.788
	C2	87.0	37.6	77.3	44.6	0.75	0.825	0.102	25.3	0.764
USP	C1	85.8	34.9	75.2	41.7	0.76	0.816	0.112	29.4	0.764
	C2	85.1	34.0	73.9	40.9	0.77	0.811	0.115	30.5	0.769
SUP	C1	88.5	41.2	79.4	48.0	0.65	0.844	0.085	28.6	0.751
	C2	88.9	41.7	80.0	48.5	0.63	0.850	0.084	29.6	0.748

The metrics show that, in general, the experiment with the best results is Experiment 4, which automatically generated scribbles only with the building detection module. This result indicates that the features in the places where the automatic scribble generator drew the scribbles provided more meaningful information than the human scribbles for the network.

In an in-depth analysis of this event, even though in the previous section, the SVM classifier learned a better representation from the human scribbles since our network does not directly associate the clusters with the supervised classes, even with a more significant error regarding the ground truth, the automatic scribbles provided a more meaningful result for the network. This might be because scribbles in the same class in an image will be assigned to the same cluster. Still, in the following image, even if the provided scribbles are in the same supervised class, their cluster might differ depending on the features of their locations. Therefore, this would explain why the pre-selected areas of the automatic scribbles provide more information and why the correct supervised classes in the automatic scribbles offer only a limited amount of data for the network. But to prove that this information does matter, the experiment with only human annotated scribbles had better results than any of the experiments, as expected.

Finally, by analyzing the remaining results, we can see the best results in the networks trained with the building detection module, proving that it is a meaningful addition to the pipeline. As for why the Unsupervised results had better metrics than the experiments, this can be explained because it had much more information to train the network since the superpixels were scattered in the entire image. But interestingly, the

IoU and F1-score of the unsupervised experiment were worse than the other experiments, indicating that using scribbles could help the network performance.

Ideally, the information of the superpixels and scribbles should be combined: either by using both as different sources of supervision and combining their losses or by combining the class of the superpixels that cross a scribble, providing a hybrid form of ground truth while keeping a single loss. We did not perform these experiments in this work, leaving this analysis for a future contribution.

Some segmentation results can be seen in Figure 19. Here we select only the best classifier for each experiment due to the lack of space to show both classifiers for every image.

In these segmentations, we can see that the network did not classify any pixel as road, and the same was perceived for buildings. This does not mean that the network cannot segment such classes but that the class mapping cannot allocate clusters to these supervised classes. This is perceived in the first and last images, where the road is segmented but not assigned in its respective class.

Regarding the experiments, almost all of them could segment grass and trees, the most common classes in the dataset. But as for water, most classifiers had difficulty. The only one that could adequately assign a river was experiment two, which used human-made scribbles to train the SVM and classify the auto-generated scribbles. This indicates that either the scribble generator is not creating sufficient scribbles in water or that the SVM is not learning this class properly. Overall, the segmentations are well-defined but lack a proper translation to the actual classes.

5.3 Applications

In this section, we test two different hypothesis regarding the capabilities of our proposed network, the first one being: "is our network capable of learning intrinsic differences between different types of terrain, in a way that it can clearly identify the border between them?". For that, we choose an specific rural dataset composed of several terrain types. The second hypothesis is: "can our network properly segment areas with a high variance and lots of elements?". Contrary to the first experiment, that evaluates the behavious in homoneous regions, in this case we want to properly extract information of highly mixed areas. For that, we choose an urban dataset with several semantic artifacts.

For the remaining of the section, we report the results for these two applications of our network in real problems. Our network proposes to be able to extract meaningful semantics and apply them to issues in an unsupervised manner so that no data labeling is required. We selected these two applications because they provide challenging problems to

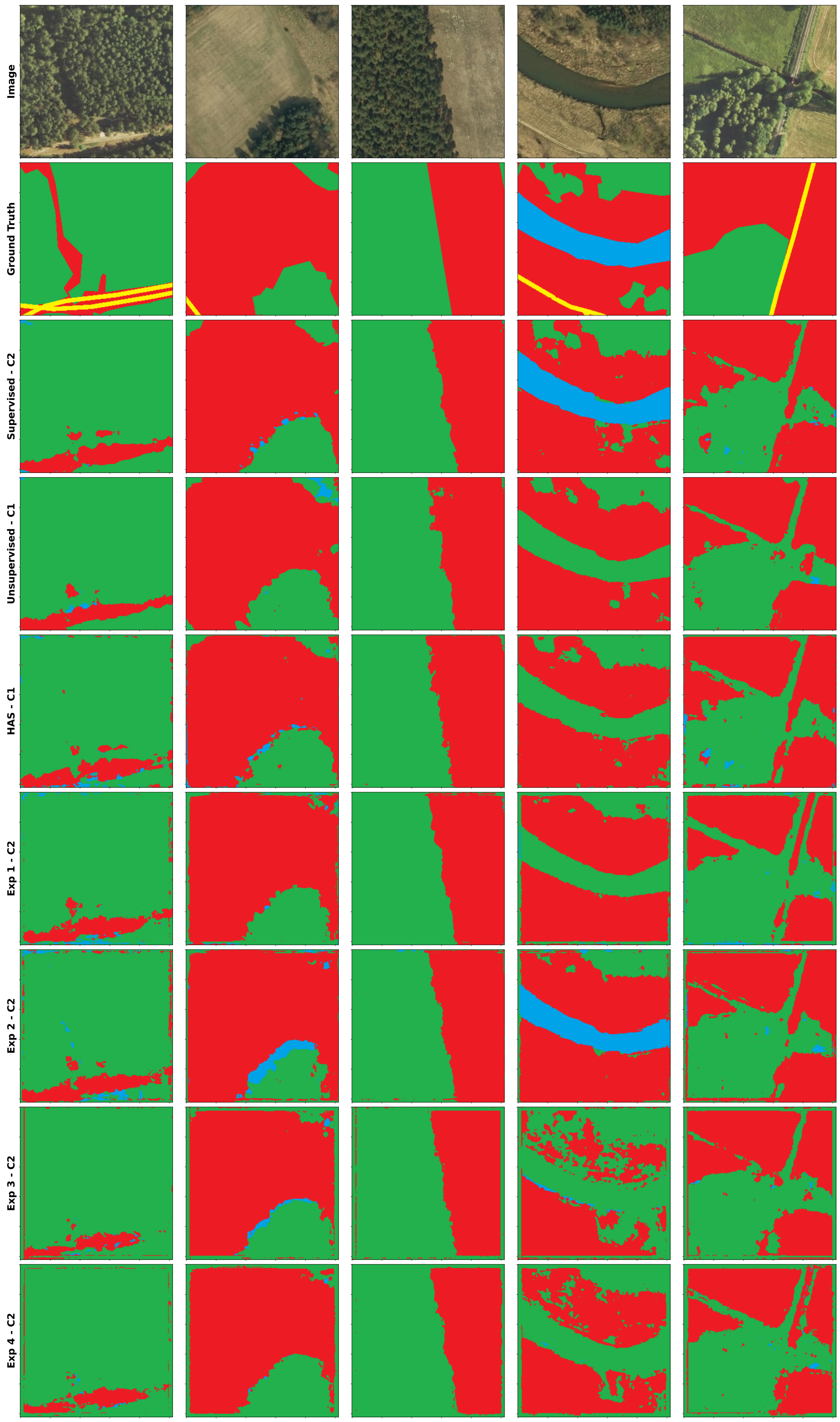


Figure 19 – Network predictions for different weak labels experiments.

test the network capacity, each using a different dataset focused on a diverse landscape. Note that although we provide quantitative results, these applications do not have a baseline for comparison with other methodologies; therefore, we focus mainly on qualitative results.

5.3.1 Border detection in rural areas

In this experiment, we try to detect the borders of different types of terrain in a rural area. This could be useful in several applications, such as mapping the region and helping in a position estimation problem. We use a rural dataset from a part of São Carlos, in Brazil [Jaimes et al., 2022], described in Section 4.3.

For this dataset, we only had the map image containing the entire region and the labels for the whole map. Both can be viewed in Figure 20. The labels for this dataset were initially obtained by segmenting the map into superpixels and assigning a class to each superpixel, so we expect the labels to have a high amount of noise. It also had 14 classes, but [Jaimes et al., 2022] proposed reducing the dataset to 4 classes; therefore, we use this condensed version. Notice that some classes do not contain only their class-name semantics (i.e., the river class also has stones), which is a direct effect of concatenating the classes. The color-label association for this dataset is:

- River: White
- Soil: Green
- Grass: Yellow
- Forest: Red

To train our network, we proposed to split the dataset by cropping random rectangles of 256 × 256 pixels. But to guarantee that we would not select the same squares for training and testing, we first divide the dataset into a 10 × 10 grid and use each half of the grid to select 1000 images. We take 20 random crops in each rectangle to obtain a fair amount of semantics. And to have more semantics in the image, we also collect the crops with 512 × 512 pixels and resize them to 256 × 256. The grid used to crop the dataset is present in Figure 21.

We train our network using our proposed hyperparameters from Section 5.1.5, using batches of 8 images and for 100 epochs. This time, we also enabled data augmentation for the training since we did not meant to compare results with other methodologies. Note that this experiment was performed before the data augmentation experiment in Section 5.1.4, therefore at the time it was still valid. In Table 11, we show the metrics for this dataset.

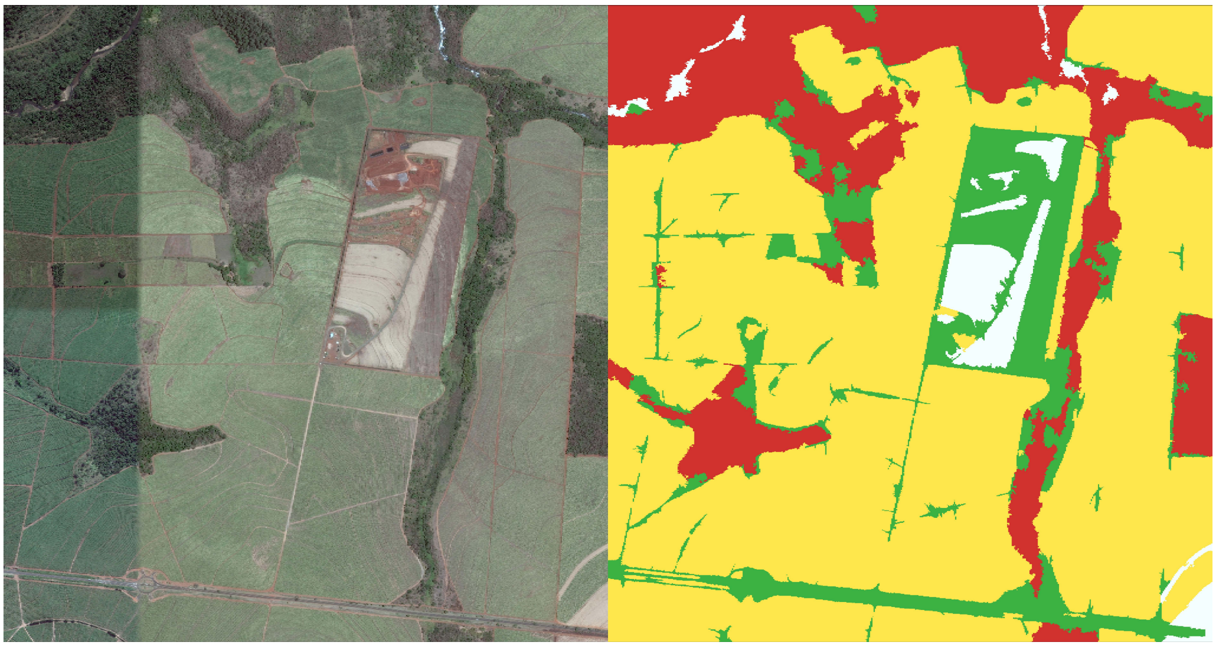


Figure 20 – Map from the São Carlos dataset and the ground truth of the four classes.

Source: produced by the author.

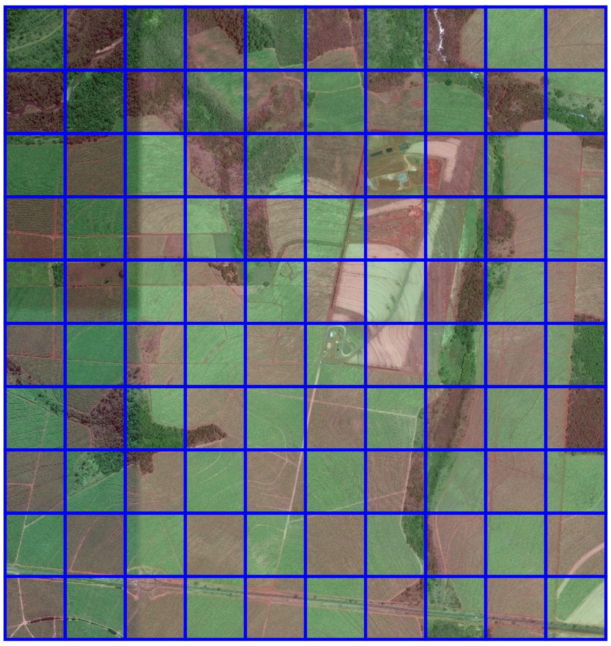


Figure 21 – Train and test division of the map from the São Carlos dataset.

Source: produced by the author.

Table 11 – Metrics for the São Carlos dataset.

C	Acc↑	IoU↑	wIoU↑	F1↑	VoI↓	PRI↑	GCE↓	BDE↓	SC↑
C1	81.5	61.7	69.2	75.6	0.83	0.789	0.109	35.0	0.764
C2	82.4	62.1	70.6	75.9	0.79	0.802	0.106	31.1	0.775

As we can see from the supervised metrics, the dataset seems to have learned the segmentation quite well. To check that, in Figure 22, we plot some of the obtained segmentations compared to the ground truth.

We can see that the ground truth itself, for most images, contained a high amount of noise. Still, it can be beneficial since introducing this noise acts as a form of regularization for the network. In the third image, it is interesting to see how the earth road in the ground truth contained noise, and the semantic segmentation fixed it in the output. The same for the image with the asphalt road, where the network drastically improved the amount of semantics compared to the ground truth by assigning a different class for asphalt, soil, and grass. As for the border regions, most of them were adequately segmented by the algorithm, although it did have some problems in the forest class, where there is a high amount of variance in color and texture.

Indeed, the model learned how to perform an accurate semantic segmentation according to the visual results, but this still does not tell us how well it learned to map the semantics. For that, we introduce two new plots.

For the first plot, we take the softmax output of every pixel for every channel and concatenate these probabilities, resulting in a volume of $Image_{width} \times Image_{height} \times K$, where K is the number of clusters. This is the same size as the output of our network. Then, when we visualize each channel, we can see the probabilities activated for that channel and check which type of semantics it mapped.

The second plot that we present is normalization for each channel individually. For that, we take each of the K channels of the output and perform a simple min-max normalization in the $Image_{width} \times Image_{height}$ pixels. The result is similar to the softmax plot, with the difference that the smaller activations that might achieve a lower probability because of the softmax normalization will better represent their value and be appropriately compared only with this channel. We plot some examples of both types of normalization in Figures 23, 24, and 25, along with the original image and their semantic segmentation, colored by the mean color of all pixels in each class. In the image names, "Act" stands for the default softmax activation, "Norm Act" stands for the normalization by channel, the number represents the output channel among the K channels, and the class shown is the ground truth class associated with this activation in the mapping step. However, in this analysis, this is not important since we are only interested in the semantics learned. Note that since we use $K = 100$, it would be unfeasible to show every channel; therefore, we

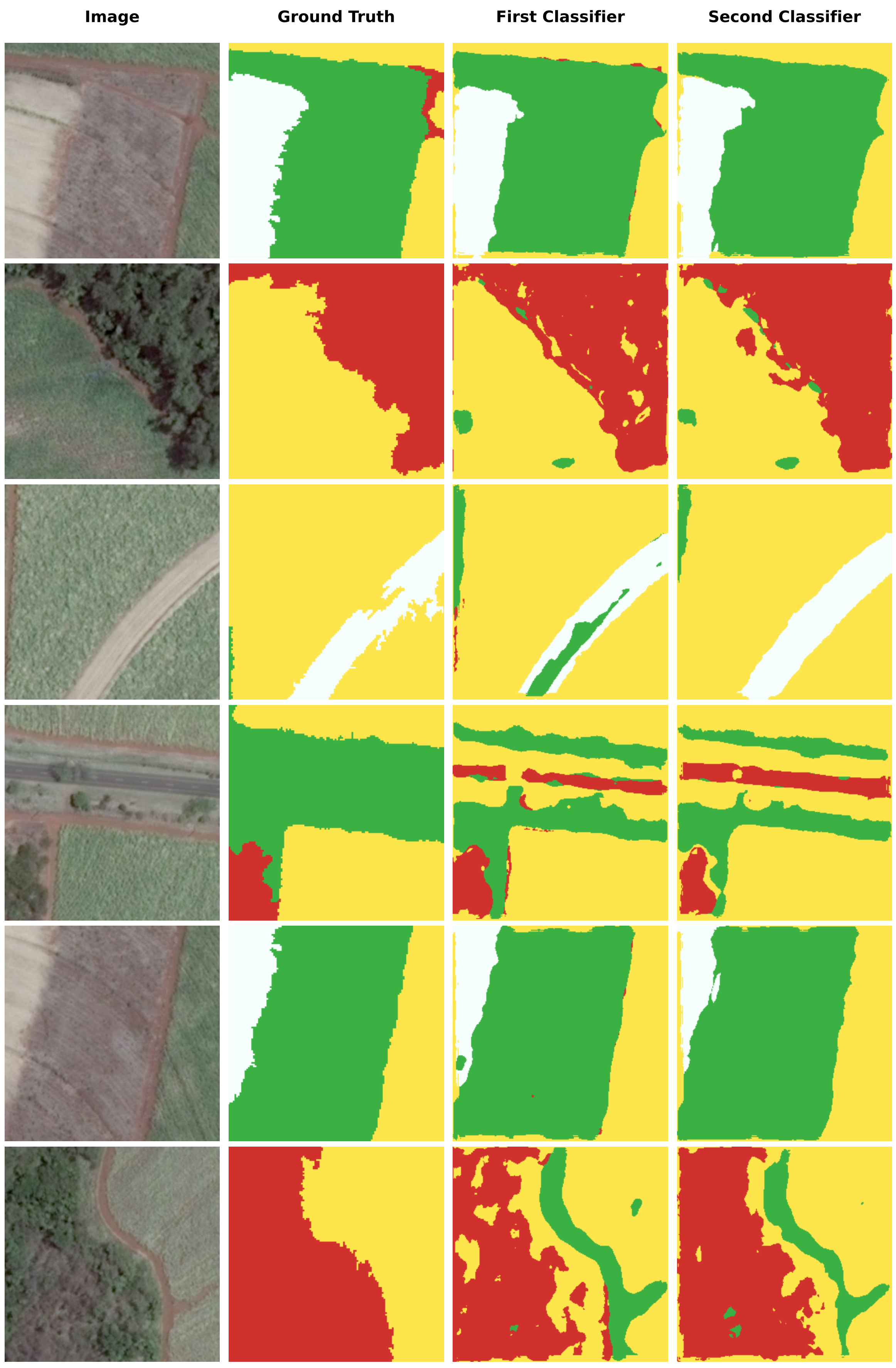


Figure 22 – Examples of semantic segmentations for the São Carlos dataset.

Source: produced by the author.

choose only a few channels with relevant semantics.

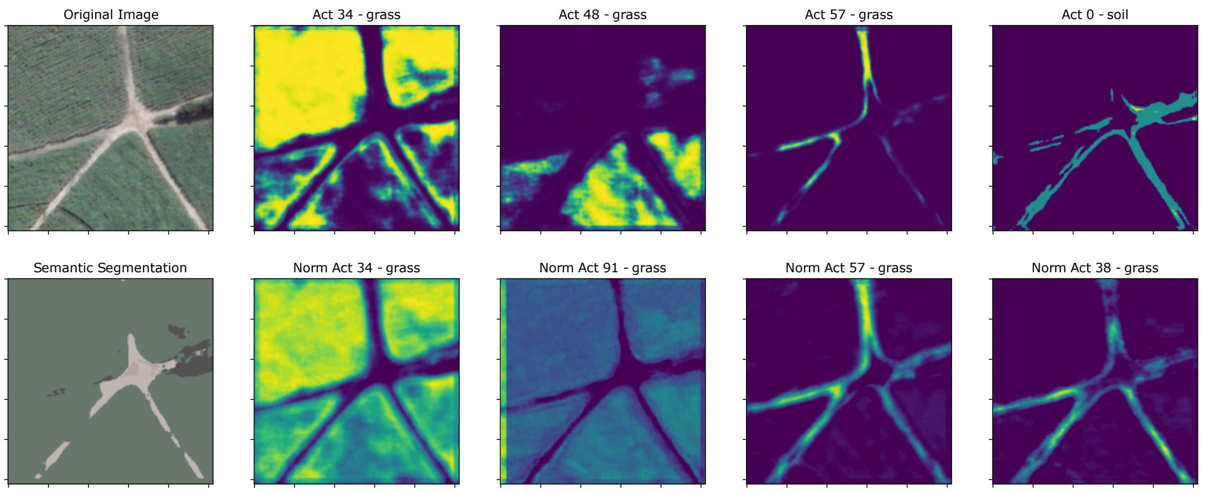


Figure 23 – Example 1 of activations for the São Carlos dataset.

Source: produced by the author.

We can see in Figure 23 that some of the softmax activations were different for different grass textures, but they complement themselves. The same can be said for the roads that complement each other, forming the entire street's connection. As for the normalized output, we can see that some activated the whole grass differently, but both could properly segment only the grass. And the same can be said for the roads.

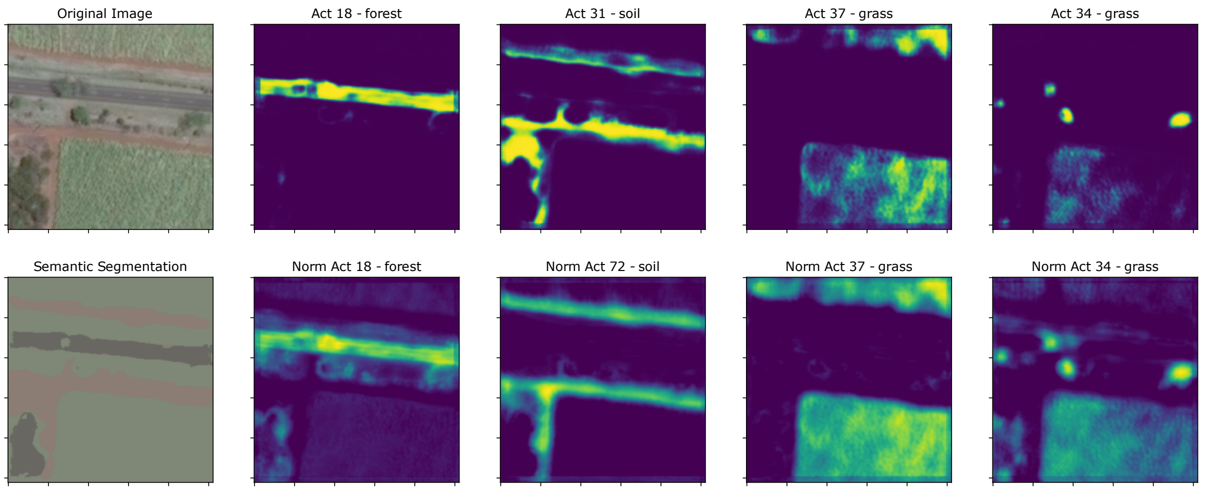


Figure 24 – Example 2 of activations for the São Carlos dataset.

Source: produced by the author.

In Figure 24, the softmax activations provided a variety of semantic objects, such as the asphalt road, the grass, the soil, and even trees. The normalized output suggests a similar separation, but note that the trees also activate for grass, showing that it is interesting to check both plots for a complete semantics analysis.

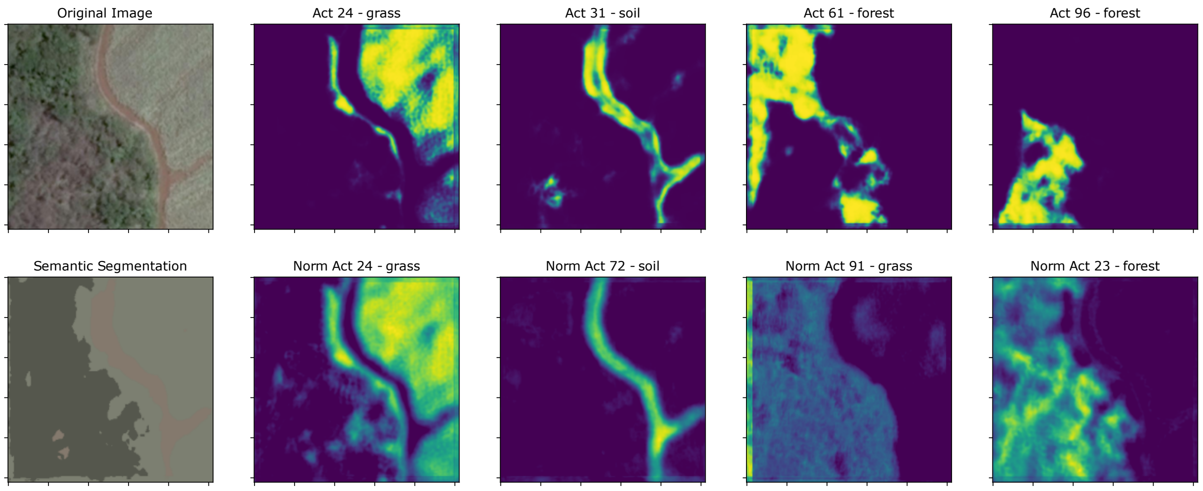


Figure 25 – Example 3 of activations for the São Carlos dataset.

Source: produced by the author.

In Figure 25, the pattern is similar to the previous images, but here we can see more clearly how the model has difficulty with the forest, which contains a wider variety of semantics. In this case, the softmax activations complement each other, while some of the normalization activations were able to represent the forest almost entirely.

As the results showed, the network could segment most regions of the image accurately, proving that the model can be used to segment the border of the images. In the activation analysis, we can see that various semantic objects were highlighted, even more than the supervised semantic mapping could describe. In a future step, these more meaningful semantic objects could be grouped and associated with more meaningful labels using robust semantic descriptors, such as natural language processing algorithms, providing a powerful tool to analyze and segment images.

5.3.2 Division of urban areas

In this second experiment, we used an urban dataset to analyze how well the model behaves in a city with a large amount of semantics and a high variance of both texture and color. We used an image of the city of Campinas, in Brazil, which only had the colors IR-R-G, as explained in Section 4.3. The whole map can be seen in Figure 26.

This dataset did not have a complete ground truth. The labels for the nine classes were all provided as weak labels, either in polygons or single pixels. Therefore, for this dataset, we do not perform the segmentation quality metrics, only the supervised metrics, since the former are not very representative in this case. The metrics can be seen in Table 12.

The dataset contained 1028 cropped images with 224×224 pixels, from which we split into 822 images for training and 206 for testing. However, in practice, for this

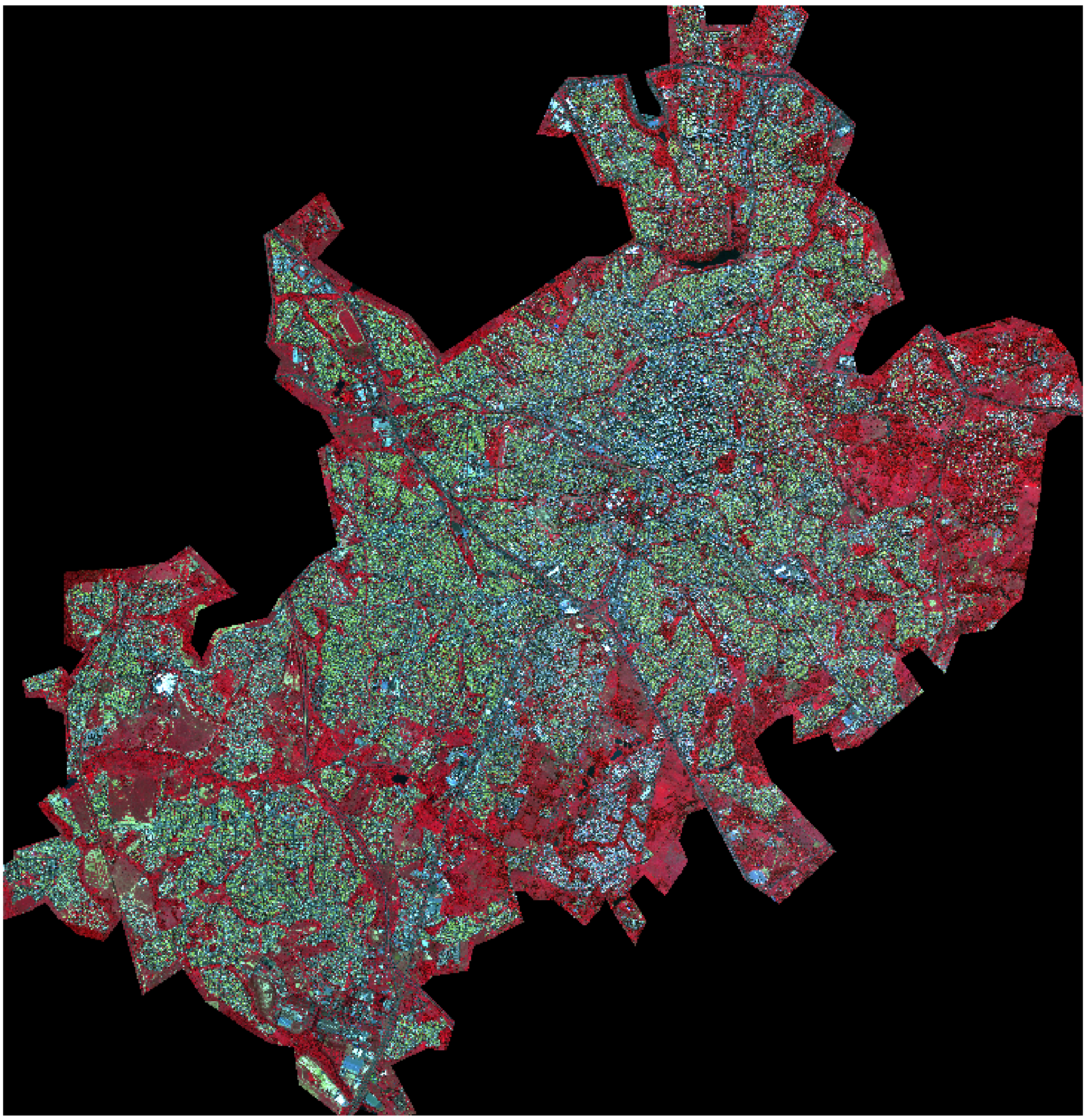


Figure 26 – Complete map of the Campinas dataset.

Source: produced by the author.

Table 12 – Metrics for the Campinas dataset.

C	Acc↑	IoU↑	wIoU↑	F1↑
C1	90.4	60.4	84.2	69.7
C2	90.4	60.7	84.0	70.8

experiment, we only used the actual classes to map the activated channels with a supervised semantic object since we did not extract the metrics. We used the same network with the same hyperparameters as in the previous application to train, also with batches of 8 images and for 100 epochs. Then we perform an analysis of the semantics in Figures 27, 28, and 29, similar to the previous application, using both the softmax normalization and channel normalization. We do not perform the supervised segmentation analysis since it would not be meaningful in this case.

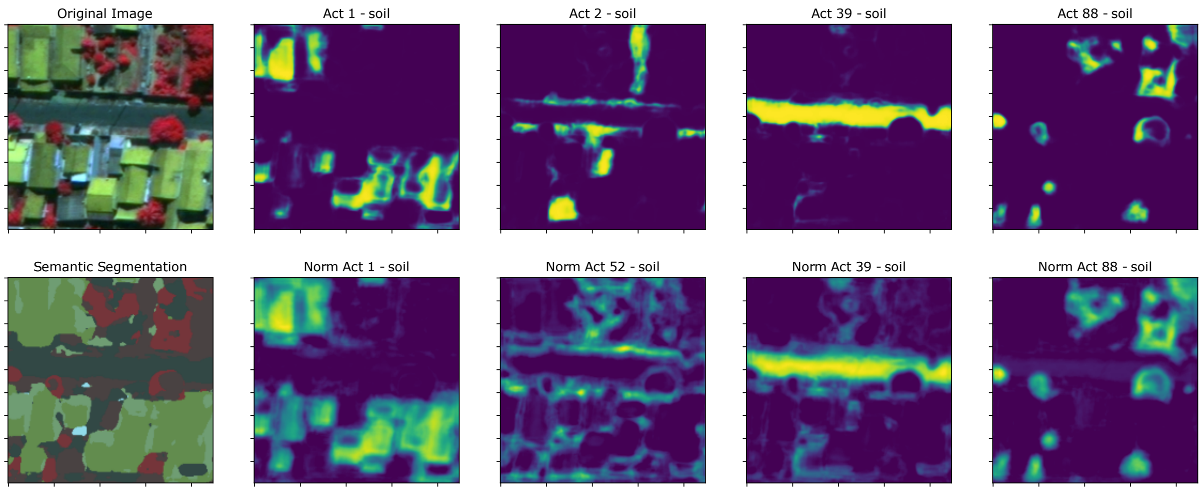


Figure 27 – Example 1 of activations for the Campinas dataset.

Source: produced by the author.

In Figure 27, we can see that the first normalized activation detected almost all buildings, even though the softmax activation couldn't. One of the activations was able to properly segment roads, vegetation, and even a sidewalk. This proves that the network clusters could learn and segment different types of semantics.

From Figure 28, we see that the network was able to learn how to segment both asphalt and sidewalks. In the sidewalk image, the network also detected the pedestrian crosswalk and the normalized output saw more details than the softmax output. The network was also able to recognize houses and vegetation.

And in Figure 29, the network could segment different types of buildings, along with roads and vegetation. In this case, the normalized activations also carried more information than the softmax activations.

These previous analyses show that the network can adequately segment individual objects in the images. But we performed a second experiment to analyze if it could segment

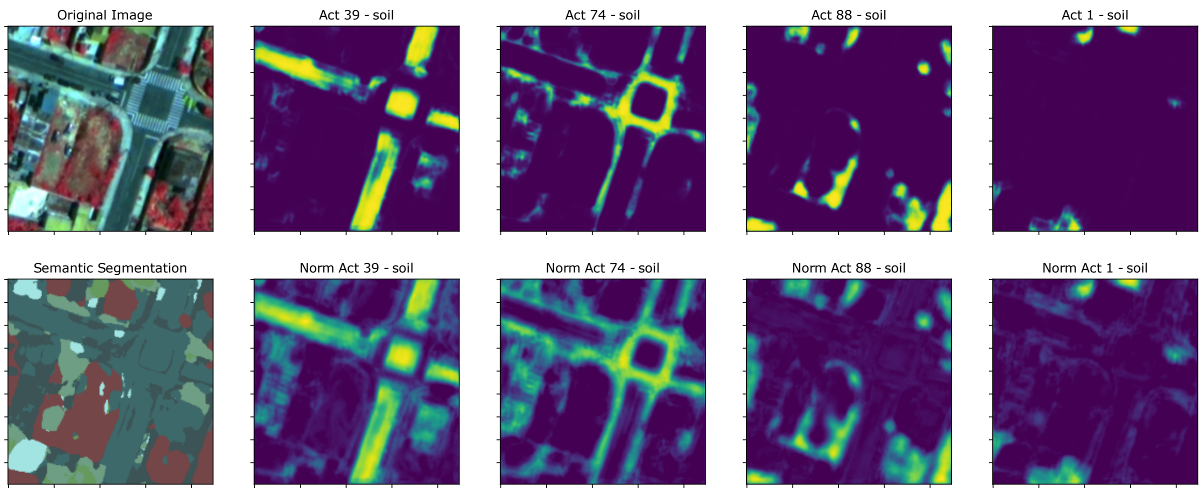


Figure 28 – Example 2 of activations for the Campinas dataset.

Source: produced by the author.

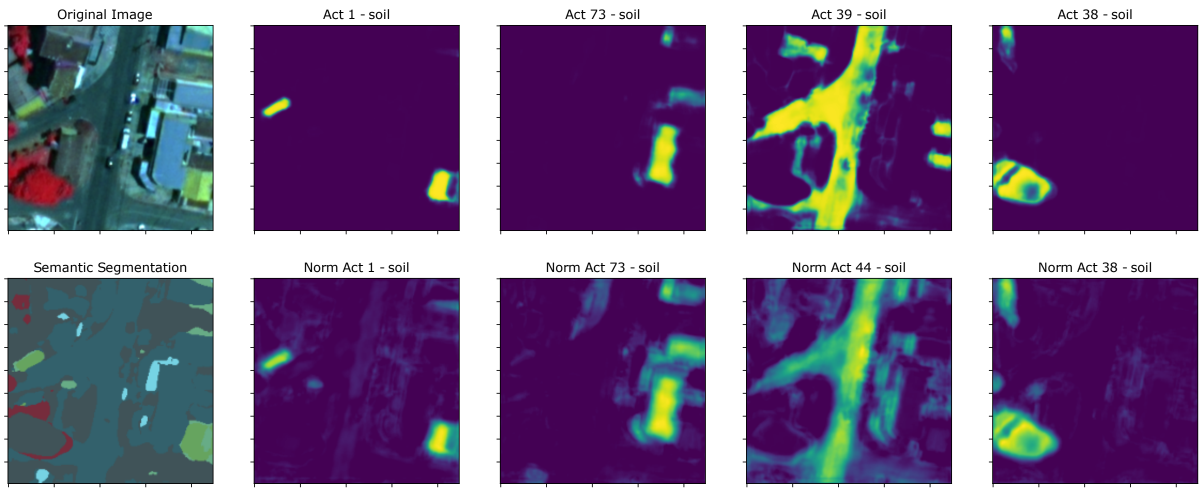


Figure 29 – Example 3 of activations for the Campinas dataset.

Source: produced by the author.

and divide an urban area into various segments. Here we cropped a square with an area of $1km^2$ in the center of the map, resulting in an image with 11111×11111 pixels. The idea was to segment the whole area and see what the activations would tell us. Since it is a large image, we have a memory limitation when performing the analysis; therefore, we conduct a downsampling of 10 times the size of this image, resulting in an image of 1111×1111 pixels. But to make a more fair downsampling, we perform it in the feature space. But first, we verify if a downsampling of 10 times would result in a massive loss of information. Both the cropped image and the same image with bilinear downsampling as shown in Figure 30.

The figure shows that there is not much loss of information since both images are still almost identical. Then we proceed to perform the downsampling in the feature space. First, we divide the image into segments of 100×100 pixels and feedforward these

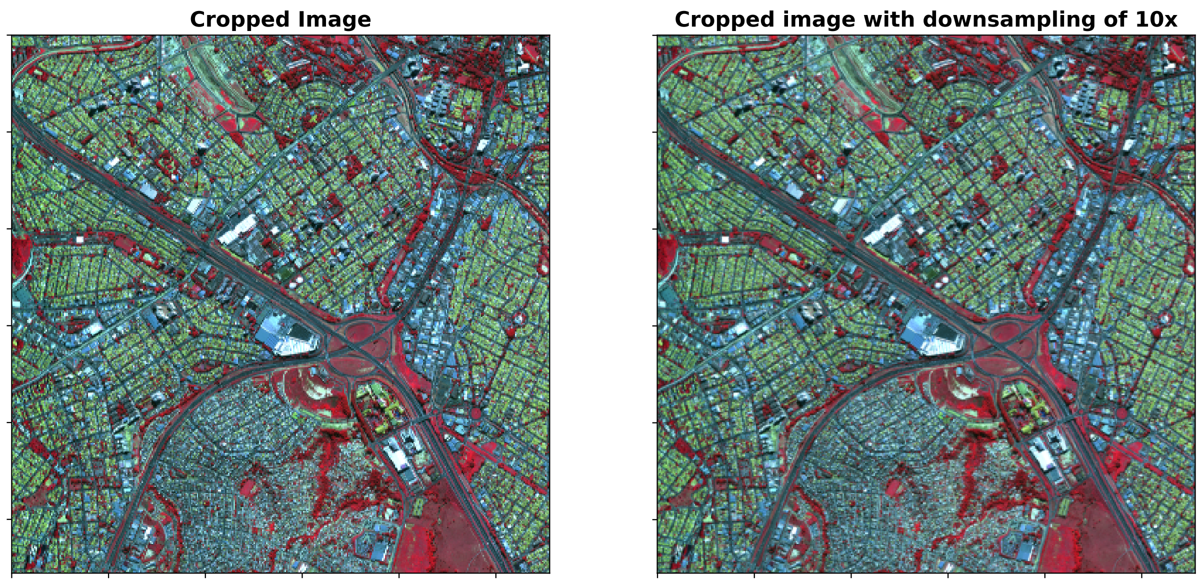


Figure 30 – Downsample comparison of a cropped area of the Campinas dataset.

Source: produced by the author.

squares through the model. We then take the mean of the features of each pixel for each channel, resulting in a $1 \times 1 \times K$ output. This way, we can reduce the size of the image by 10 while still performing a relevant analysis since, in a large image, there is not much loss of information even using a large window of 100 pixels.

We perform both the softmax and channel normalization analysis with this resulting image. The results of a few channels can be seen in Figures 31, 32, 33, 34, and 35.

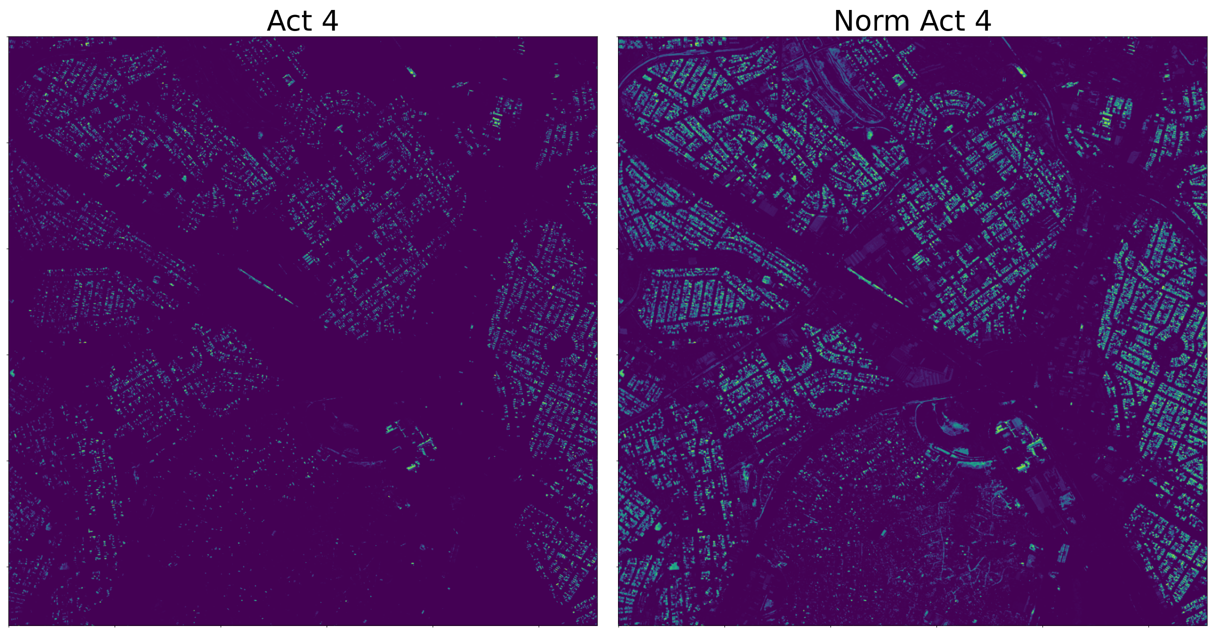


Figure 31 – Example 1 of activations of a cropped area of the Campinas dataset.

Source: produced by the author.

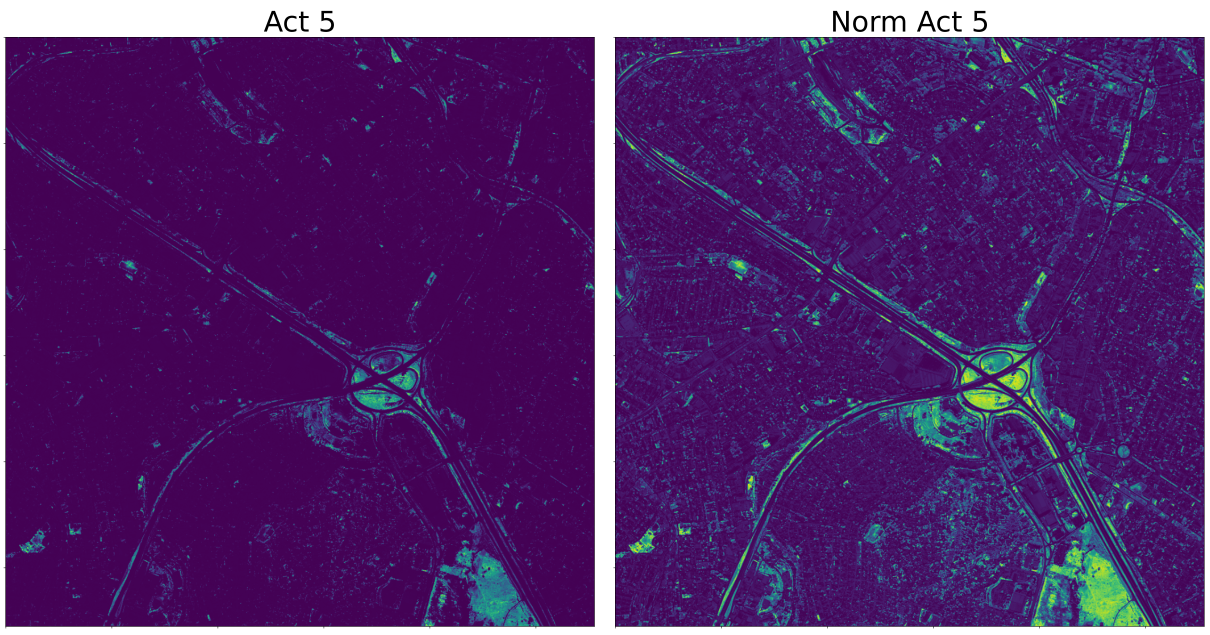


Figure 32 – Example 2 of activations of a cropped area of the Campinas dataset.

Source: produced by the author.

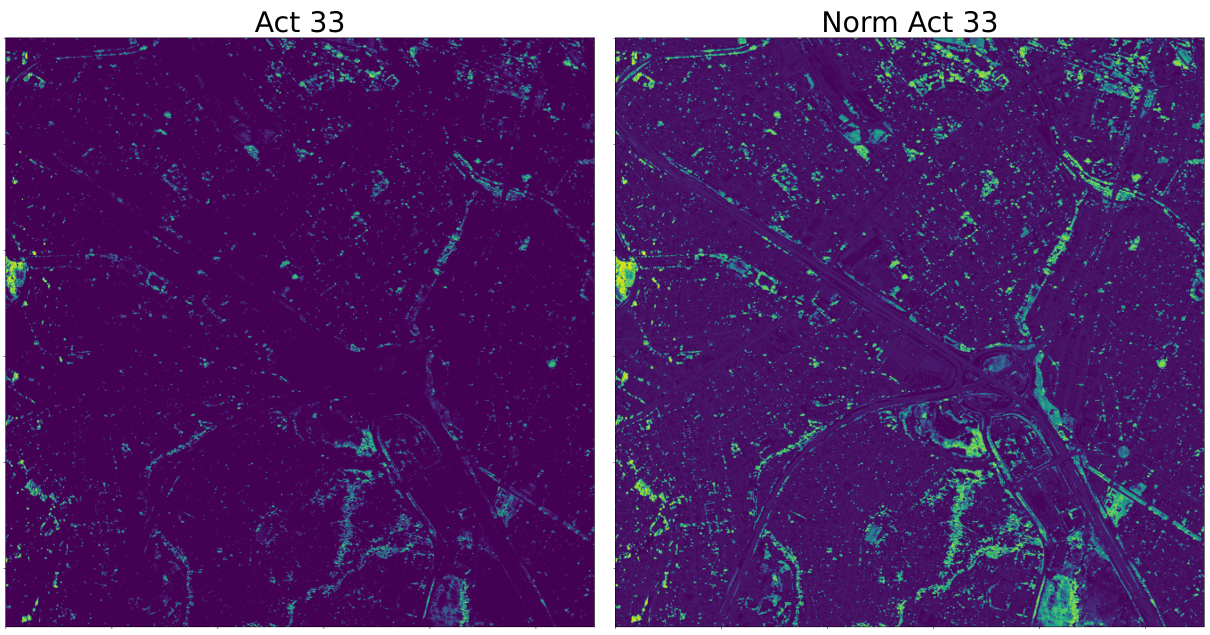


Figure 33 – Example 3 of activations of a cropped area of the Campinas dataset.

Source: produced by the author.

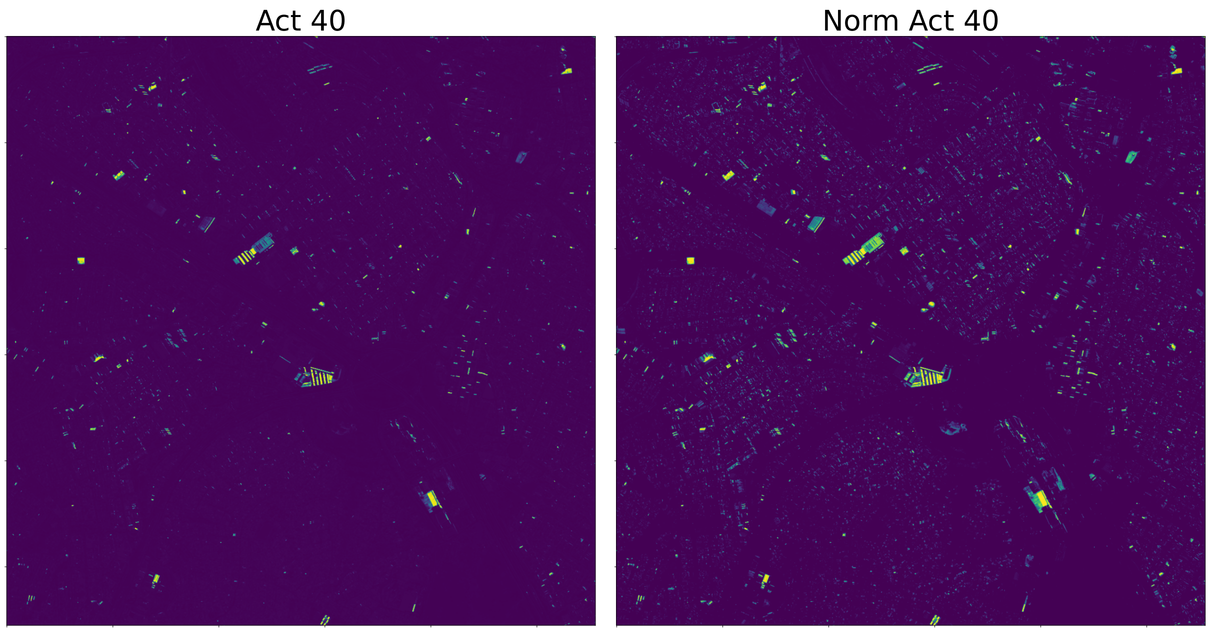


Figure 34 – Example 4 of activations of a cropped area of the Campinas dataset.

Source: produced by the author.

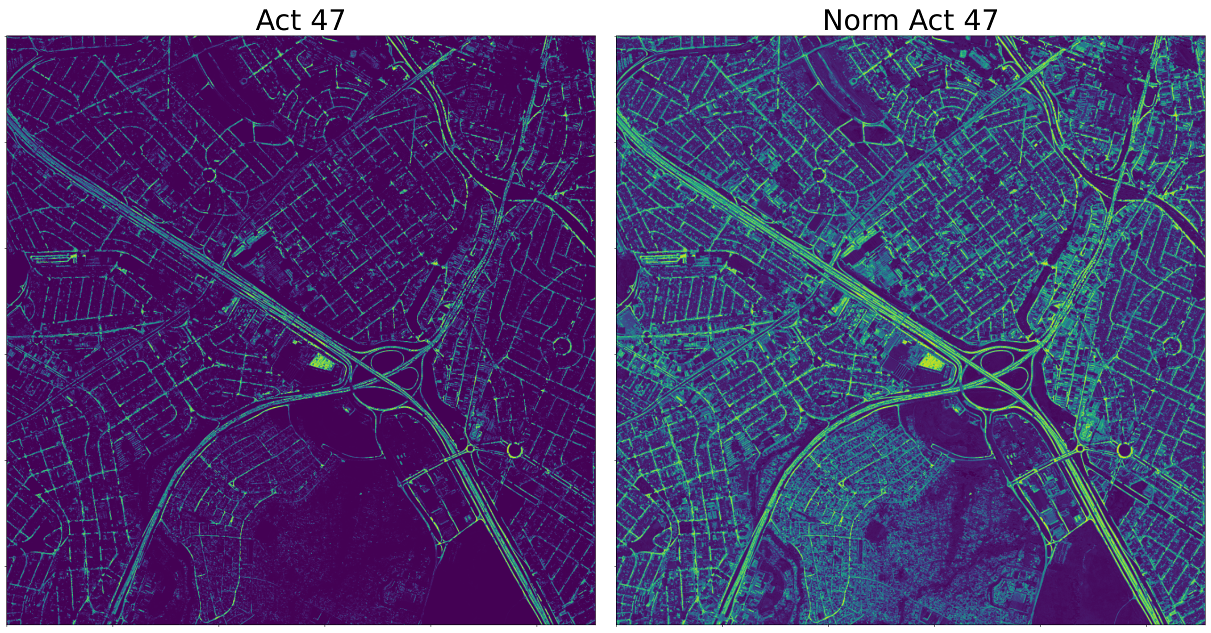


Figure 35 – Example 5 of activations of a cropped area of the Campinas dataset.

Source: produced by the author.

These figures show how the model can adequately recognize different semantic types in an urban area. Some activations can segment the asphalt roads, as in Figure 35, another can segment most houses in the selected region, as seen in Figure 31, or even specific types of houses, as shown in Figure 34, and the network can even differentiate grass, in Figure 32, and trees, in Figure 33. Overall, this experiment shows that this network can be used to segment pieces of semantic elements in an urban area in an efficient manner, being an excellent tool for applications that require an analysis of elements in a city.

Chapter 6

Conclusion

This work proposes an algorithm for a fully unsupervised semantic segmentation methodology. First, we segmented the pipeline from Kanezaki [2018] into blocks. For each block, we performed experiments with variations obtained from a state-of-art review of every CNN-based solution in the literature. This way, we were able to improve existing methodologies and create a final solution that recycles knowledge and contains the best parts of the previous state-of-art solutions. Our pipeline is also generalist, meaning it can take different input types to guide the learning process, such as unsupervised, supervised, and weakly supervised.

In addition to the contributions presented, other aspects of the pipeline were investigated in this work. Along with the results we reported for the feature extractor, we tried to adapt the network with internal changes, such as increasing the kernel size of the filters to 5×5 or exchanging the upsampling layer with a transpose convolution. But none of these changes improved the results, so we decided that changes to the feature extractor should be analyzed in a future continuation of our work.

For the loss function, one of the main reasons that motivated us to use two classifiers was to use a loss that could manage the information from both classifiers, as in papers exploring mutual information losses. We adapted the mutual information maximization loss from Harb and Knöbelreiter [2021] but had no success improving the results. Therefore, we also did not report this experiment, leaving this investigation with potential new losses as a suggestion of continuity of this work.

Finally, our main contributions are combining multiple solutions into one and improving these methodologies to extract the best of them. The results we achieved in a common dataset in literature are competitive with (and, in some cases, even surpassed) previous studies that have adopted the same research line with CNN-based descriptors. Even though it did not reach the novel visual transformers technology, the extracted semantics' results and complexity are still beneficial and applicable to various remote sensing tasks.

We also proposed an automatic scribble generator methodology. We supposed that regions with similar content in images could be grouped into areas and that by using an interpolation method, we could draw scribbles. We tested this methodology and achieved good results, correctly generating scribbles in several regions with contexts.

To accompany the scribble generator, we proposed a scribble classifier methodology. By manually assigning classes to a small set of scribbles, using a tool that we provide, a large dataset of images with scribbles could be automatically classified. The classifier results showed that using our proposed features, most scribbles could be assigned to their actual classes accurately.

We tested the training of our network using only scribbles and compared the results under several conditions, concluding that our automatically generated scribbles are fitted to be used as a source of supervision for our model, achieving metrics similar to fully supervised results. Our main contributions can be summarized as the novel pipeline for scribble generation, the scribble classifier methodology, and the editing tool that allows a variety of operations in scribbles.

We also proposed two applications. In the first one, related to the segmentation of rural areas into different regions, our pipeline could recognize the different types of semantics in remote sensing images, finding and segmenting a variety of terrains, therefore, being able to find the borders between them.

And in our second application, we tested our network in an urban environment, where we checked the network's ability to segment several parts of the city. The network was able to find specific semantic objects in a city properly and to segment the elements of a larger image containing several neighborhoods and streets, such as the roads, the vegetation, and some types of rooftops, providing meaningful insights about the city, proving that a methodology is a powerful tool for analyzing semantics.

Overall, this work contributes to the literature with different propositions by providing a practical unsupervised semantic segmentation methodology that can be applied in various remote sensing applications.

For future works, in the semantic segmentation pipeline we could explore a loss that uses the information from both classifiers, so that we can fully take advantage of them being trained to classify the same images with similar classes. We could also perform some ablation studies in the feature extractor, deeply exploring the contributions of every block, and even changing meaningful parameters inside the blocks. We could also test bigger architectures from literature, while trying to perform transfer learning from similar remote sensing problems. And we could also try to migrate some of our ideas to a Visual Transformer architecture, since those are the current state-of-art for this kind of problem.

As for the scribble generator pipeline, in future works we could explore other methodologies for creating and selecting the superpixels, that maybe solve the problem of classes with small and narrow superpixels whitout the need for an external classifier. We could also evaluate the benefits of approximating the superpixel centers to human scribbles, since in practice we are using the features of the entire superpixel area. And we could also perform some ablation studies in every step of the pipeline, to further evalueate their fully contribution to the overall results.

And for the applications, in future works we could further explore even use cases of our model, in any remote sensing problems that may require the analysis of the terrain and its semantics. Some examples include deforestation and wildfire detection, terrain delimitation, urban planning, and even specific semantic objects detections (e.g. building and water bodies detection).

References

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2281, 2012. ISSN 01628828. doi: 10.1109/TPAMI.2012.120. URL http://lcavwww.epfl.ch/$\sim$achanta/%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.8269&rep=rep1&type=pdf%5Cnhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.8269&rep=rep1&type=pdf. 37

E. Alfonseca, K. Filippova, J. Y. Delort, and G. Garrido. Pattern learning for relation extraction with a hierarchical topic model. *50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference*, 2:54–59, 2012. 27

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. ISSN 01628828. doi: 10.1109/TPAMI.2010.161. 51

V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2644615. 19

M. Barthakur and K. K. Sarma. Semantic Segmentation using K-means Clustering and Deep Learning in Satellite Image. *Proceedings of 2nd International Conference on Innovations in Electronics, Signal Processing and Communication, IESC 2019*, pages 192–196, 2019. doi: 10.1109/IESPC.2019.8902391. 23, 33

D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. Interactively co-segmentating topically related images with intelligent scribble guidance. *International Journal of Computer Vision*, 93(3):273–292, 2011. ISSN 09205691. doi: 10.1007/s11263-010-0415-x. 28, 39, 43

D. Berend and A. Kontorovich. Consistency of weighted majority votes. *Advances in Neural Information Processing Systems*, 4(January):3446–3454, 2014. ISSN 10495258. 27

P. Bhatt, S. Sarangi, and S. Pappula. Unsupervised image segmentation using convolutional neural networks for automated crop monitoring. *ICPRAM 2019 - Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, pages 887–893, 2019. doi: 10.5220/0007687508870893. 23, 33, 37

A. Boguszewski, D. Batorski, N. Ziemba-Jankowska, T. Dziedzic, and A. Zambrzycka. LandCover.ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1102–1110, 2021. ISSN 21607516. doi: 10.1109/CVPRW53098.2021.00121. 53, 72

J. Bootkrajang and A. Kabán. Label-noise robust logistic regression and its applications. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7523 LNAI(PART 1):143–158, 2012. ISSN 03029743. doi: 10.1007/978-3-642-33460-3_15. 27

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11218 LNCS:139–156, 2018. ISSN 16113349. doi: 10.1007/978-3-030-01264-9_9. URL `http://dx.doi.org/10.1007/978-3-030-01264-9_9`. 19, 21

J. Castillo-Navarro, B. Le Saux, A. Boulch, N. Audebert, and S. Lefèvre. Semi-supervised semantic segmentation in Earth Observation: the MiniFrance suite, dataset analysis and multi-task network study. *Machine Learning*, 2021. ISSN 15730565. doi: 10.1007/s10994-020-05943-y. 28

F. J. Chang, Y. Y. Lin, and K. J. Hsu. Multiple structured-instance learning for semantic segmentation with uncertain training data. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 360–367, 2014. ISSN 10636919. doi: 10.1109/CVPR.2014.53. 27

L. C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to Scale: Scale-Aware Semantic Image Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:3640–3649, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.396. 19

L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40 (4):834–848, 2018. ISSN 01628828. doi: 10.1109/TPAMI.2017.2699184. 19

J. H. Cho, U. Mall, K. Bala, and B. Hariharan. Picie: Unsupervised Semantic Segmentation using Invariance and Equivariance in Clustering. *Proceedings of the IEEE Computer*

Society Conference on Computer Vision and Pattern Recognition, pages 16789–16799, 2021. ISSN 10636919. doi: 10.1109/CVPR46437.2021.01652. 22, 61

J. Clarke, D. Goldwasser, M. W. Chang, and D. Roth. Driving semantic parsing from the world's response. *CoNLL 2010 - Fourteenth Conference on Computational Natural Language Learning, Proceedings of the Conference*, pages 18–27, 2010. 27

J. Dai, K. He, and J. Sun. Instance-Aware Semantic Segmentation via Multi-task Network Cascades. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:3150–3158, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.343. 27

X. Ding, Y. Xu, L. Deng, and X. Yang. Colorization using quaternion algebra with automatic scribble generation. *Lecture Notes in Computer Science*, 7131 LNCS:103–114, 2012. ISSN 03029743. doi: 10.1007/978-3-642-27355-1_12. 28, 39, 40

G. Druck, B. Settles, and A. McCallum. Active learning by labeling features. *EMNLP 2009 - Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: A Meeting of SIGDAT, a Special Interest Group of ACL, Held in Conjunction with ACL-IJCNLP 2009*, pages 81–90, 2009. doi: 10.3115/1699510.1699522. 27

M. Eliasof, N. Ben Zikri, and E. Treister. Unsupervised Image Semantic Segmentation Through Superpixels and Graph Neural Networks. *SSRN Electronic Journal*, 2022. doi: 10.2139/ssrn.4255497. 68

R. W. Fairclough, R. A. Bain, S. J. Holmes, C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, M. D. Zeiler, R. Fergus, S. Adalı, P.-y. Chen, Y. Su, D. Cai, Y. Wang, D. Vandyke, S. Baker, P. Li, N. Collier, F. Schroff, J. Philbin, S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 40(1):1–9, 2015. ISSN 10636919. 18

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. ISSN 09205691. doi: 10.1023/B:VISI.0000022288.19776.77. 23

D. Feng, C. Haase-Schutz, L. Rosenbaum, H. Hertlein, C. Glaser, F. Timm, W. Wiesbeck, and K. Dietmayer. Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, 2021. ISSN 15580016. doi: 10.1109/TITS.2020.2972974. 12

J. P. K. Ferreira, J. P. L. Pinto, and C. L. Castro. Weaklier Supervised: Semi-automatic Scribble Generation Applied to Semantic Segmentation. In *Anais Estendidos do XXXV*

Conference on Graphics, Patterns and Images (SIBGRAPI Estendido 2022), pages 84–87. Sociedade Brasileira de Computação - SBC, oct 2022. doi: 10.5753/sibgrapi.est.2022.23266. URL `https://sol.sbc.org.br/index.php/sibgrapi_estendido/article/view/23266`. 16, 40, 41, 44

J. Freixenet, X. Muñoz, D. Raba, J. Martí, and X. Cufí. Yet another survey on image segmentation: Region and boundary information integration. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2352:408–422, 2002. ISSN 16113349. doi: 10.1007/3-540-47977-5_27. URL `http://link.springer.com/10.1007/3-540-47977-5_27`. 50

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. ISSN 03401200. doi: 10.1007/BF00344251. 17

K. Guu, P. Pasupat, E. Z. Liu, and P. Liang. From Language to Programs: Bridging Reinforcement Learning and Maximum Marginal Likelihood. *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 1:1051–1062, 2017. URL `https://arxiv.org/abs/1704.07926v1`. 27

M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. T. Freeman. Unsupervised semantic segmentation by distilling feature correspondences. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=SaKO6z6Hl0c`. 21, 39, 43, 68

R. Harb and P. Knöbelreiter. InfoSeg: Unsupervised Semantic Image Segmentation with Mutual Information Maximization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13024 LNCS:18–32, 2021. ISSN 16113349. doi: 10.1007/978-3-030-92659-5_2. 22, 61, 68, 92

B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8695 LNCS(PART 7):297–312, 2014. ISSN 16113349. doi: 10.1007/978-3-319-10584-0_20. 27

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.90. 18

K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-October, pages 2980–

2988. IEEE, oct 2017. ISBN 9781538610329. doi: 10.1109/ICCV.2017.322. URL http://ieeexplore.ieee.org/document/8237584/. 19

A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *Computer Vision and Pattern Recognition*, 14(2):53–57, 2017. ISSN 15071367. doi: 10.48550/arXiv.1704.04861. URL http://arxiv.org/abs/1704.04861. 18

J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-Excitation Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):2011–2023, 2020. ISSN 19393539. doi: 10.1109/TPAMI.2019.2913372. 23, 33

Y. Hua, D. Marcos, L. Mou, X. X. Zhu, and D. Tuia. Semantic Segmentation of Remote Sensing Images with Sparse Annotations. *IEEE Geoscience and Remote Sensing Letters*, 19, 2022. ISSN 15580571. doi: 10.1109/LGRS.2021.3051053. 27

G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993. 18

T. Ilyas, A. Khan, M. Umraiz, and H. Kim. SEEK: A framework of superpixel learning with cnn features for unsupervised segmentation. *Electronics (Switzerland)*, 9(3), 2020. ISSN 20799292. doi: 10.3390/electronics9030383. 7, 23, 33, 48, 57

ISPRS. 2D Semantic Labeling Contest, 2016. URL http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html. 53

B. R. A. Jaimes, J. P. K. Ferreira, and C. L. Castro. Unsupervised Semantic Segmentation of Aerial Images with Application to UAV Localization. *IEEE Geoscience and Remote Sensing Letters*, 19, 2022. ISSN 15580571. doi: 10.1109/LGRS.2021.3113878. 38, 39, 42, 53, 59, 65, 79

X. Ji, A. Vedaldi, and J. Henriques. Invariant information clustering for unsupervised image classification and segmentation. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:9864–9873, 2019. ISSN 15505499. doi: 10.1109/ICCV.2019.00996. 20, 21, 22, 59, 61, 68

X. Jiao, Y. Chen, and R. Dong. An unsupervised image segmentation method combining graph clustering and high-level feature representation. *Neurocomputing*, 409:83–92, 2020. ISSN 18728286. doi: 10.1016/j.neucom.2020.05.073. 22

M. Jin, R. Jia, Z. Kang, I. C. Konstantakopoulos, and C. J. Spanos. PresenceSense: Zero-training algorithm for individual presence detection based on power monitoring. *BuildSys 2014 - Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 1–10, 2014. doi: 10.1145/2674061.2674073. 27

M. Jin, R. Jia, and C. J. Spanos. Virtual Occupancy Sensing: Using Smart Meters to Indicate Your Presence. *IEEE Transactions on Mobile Computing*, 16(11):3264–3277, 2017. ISSN 15361233. doi: 10.1109/TMC.2017.2684806. 27

L. Jing and Y. Tian. Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11): 4037–4058, 2021. ISSN 19393539. doi: 10.1109/TPAMI.2020.2992393. 15

A. Kanezaki. Unsupervised image segmentation by backpropagation. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2018-April:1543–1547, 2018. ISSN 15206149. doi: 10.1109/ICASSP.2018.8462533. 7, 22, 23, 24, 28, 31, 32, 33, 38, 56, 57, 92

T. W. Ke, J. J. Hwang, Y. Guo, X. Wang, and S. X. Yu. Unsupervised Hierarchical Semantic Segmentation with Multiview Cosegmentation and Clustering Transformers. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022-June:2561–2571, 2022. ISSN 10636919. doi: 10.1109/CVPR52688.2022.00260. 68

Z. Khan and J. Yang. Bottom-up unsupervised image segmentation using FC-Dense u-net based deep representation clustering and multidimensional feature fusion based region merging. *Image and Vision Computing*, 94, 2020. ISSN 02628856. doi: 10.1016/j.imavis.2020.103871. 23, 33, 37, 48, 49, 50

B. Kim and J. C. Ye. Mumford-shah loss functional for image segmentation with deep learning. *IEEE Transactions on Image Processing*, 29:1856–1866, 2020. ISSN 19410042. doi: 10.1109/TIP.2019.2941265. 22, 33, 34, 37, 48, 49, 63

W. Kim, A. Kanezaki, and M. Tanaka. Unsupervised Learning of Image Segmentation Based on Differentiable Feature Clustering. *IEEE Transactions on Image Processing*, 29:8055–8068, 2020. ISSN 19410042. doi: 10.1109/TIP.2020.3011269. 33, 63

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2: 1097–1105, 2012. ISSN 10495258. 17

S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017. 27

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998. ISSN 00189219. doi: 10.1109/5.726791. URL `http://ieeexplore.ieee.org/document/726791/`. 13, 17

P. Liang, M. I. Jordan, and D. Klein. Learning from measurements in exponential families. *ACM International Conference Proceeding Series*, 382, 2009. doi: 10.1145/1553374.1553457. 27

D. Lin, J. Dai, J. Jia, K. He, and J. Sun. ScribbleSup: Scribble-Supervised Convolutional Networks for Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:3159–3167, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.344. 27, 42

Q. Lin, W. Zhong, and J. Lu. Deep superpixel cut for unsupervised image segmentation. *Proceedings - International Conference on Pattern Recognition*, pages 8870–8876, 2020. ISSN 10514651. doi: 10.1109/ICPR48806.2021.9411968. 24, 32, 33, 34, 38, 48, 49, 51

T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:936–944, 2017. doi: 10.1109/CVPR.2017.106. 19

P. Luc, C. Couprie, S. Chintala, and J. Verbeek. Semantic segmentation using adversarial networks. *CoRR*, abs/1611.08408, 2016. URL `http://arxiv.org/abs/1611.08408`. 19

F. Ma, F. Gao, J. Sun, H. Zhou, and A. Hussain. Weakly supervised segmentation of SAR imagery using superpixel and hierarchically adversarial CRF. *Remote Sensing*, 11(5), 2019. ISSN 20724292. doi: 10.3390/rs11050512. 27

G. S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11:955–984, 2010. ISSN 15324435. 27

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 *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 416–423, 2001. doi: 10.1109/ICCV.2001.937655. 50

T. Meenpal, A. Balakrishnan, and A. Verma. Facial Mask Detection using Semantic Segmentation. *2019 4th International Conference on Computing, Communications and Security, ICCCS 2019*, 2019. doi: 10.1109/CCCS.2019.8888092. 13

M. Meilă. Comparing clusterings - An axiomatic view. In *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, pages 577–584, 2005. ISBN 1595931805. 48

A. Milioto, P. Lottes, and C. Stachniss. Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs.

Proceedings - IEEE International Conference on Robotics and Automation, pages 2229–2235, 2018. ISSN 10504729. doi: 10.1109/ICRA.2018.8460962. 13

S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022. ISSN 19393539. doi: 10.1109/TPAMI.2021.3059968. 13, 24

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, Aug. 2009. Association for Computational Linguistics. URL `https://aclanthology.org/P09-1113`. 27

S. E. Mirsadeghi, A. Royat, and H. Rezatofighi. Unsupervised Image Segmentation by Mutual Information Maximization and Adversarial Regularization. *IEEE Robotics and Automation Letters*, 6(4):6931–6938, 2021. ISSN 23773766. doi: 10.1109/LRA.2021.3095311. 68

N. C. Mithun, S. Paul, and A. K. Roy-Chowdhury. Weakly supervised video moment retrieval from text queries. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:11584–11593, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.01186. 27

A. Y. Ng and M. I. Jordan. On discriminative vs. Generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, 2002. ISSN 10495258. 13

Y. Ouali, C. Hudelot, and M. Tami. Autoregressive Unsupervised Image Segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12352 LNCS:142–158, 2020. ISSN 16113349. doi: 10.1007/978-3-030-58571-6_9. 68

C. Pantofaru and M. Hebert. A Comparison of Image Segmentation Algorithms. Technical report, Carnegie Mellon University, 2005. URL `http://repository.cmu.edu/cgi/viewcontent.cgi?article=1335&context=robotics%5Cnpapers2://publication/uuid/219D0A8C-2FB1-4CB4-AC9B-7E612F2CEA55`. 49

D. Pathak, P. Krahenbuhl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 International Conference on Computer Vision, ICCV 2015:1796–1804, 2015. ISSN 15505499. doi: 10.1109/ICCV.2015.209. 27

N. Pourian, S. Karthikeyan, and B. S. Manjunath. Weakly supervised graph based semantic segmentation by learning communities of image-parts. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 International Conference on Computer Vision, ICCV 2015:1359–1367, 2015. ISSN 15505499. doi: 10.1109/ICCV.2015.160. 27

A. Rabinovich, A. Vedaldi, and S. Belongie. Does Image Segmentation Improve Object Categorization ? *UCSD CSE Technical Report*, 2007. 13

A. Ratner, C. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. *Advances in Neural Information Processing Systems*, pages 3574–3582, 2016. ISSN 10495258. 27

A. Ratner, S. Bach, P. Varma, and C. Ré. An Overview of Weak Supervision, 2017a. URL https://www.snorkel.org/blog/weak-supervision. 26, 27

A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3): 269–282, 2017b. ISSN 21508097. doi: 10.14778/3157794.3157797. 27

Y. Roh, G. Heo, and S. E. Whang. A Survey on Data Collection for Machine Learning: A Big Data-AI Integration Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347, 2021. ISSN 15582191. doi: 10.1109/TKDE.2019.2946162. 25

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:234–241, 2015. ISSN 16113349. doi: 10.1007/978-3-319-24574-4_28. 18

B. Roth and D. Klakow. Combining generative and discriminative model scores for distant supervision. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 24–29, 2013. 27

C. Ru, J. Tang, S. Xie, S. Li, and T. Wang. Reducing wrong labels in distant supervision for relation extraction. *Guofang Keji Daxue Xuebao/Journal of National University of Defense Technology*, 40(3):148–152, 2018. ISSN 10012486. doi: 10.11887/j.cn.201803023. 27

P. Ruvolo, J. Whitehill, and J. R. Movellan. Exploiting Commonality and Interaction Effects in Crowdsourcing Tasks Using Latent Factor Models. *Advances in neural information processing systems*, pages 1–9, 2015. 27

S. Sah. Machine Learning: A Review of Learning Types. *ResearchGate*, July 2020. URL www.preprints.org. 25

S. Saha, S. Sudhakaran, B. Banerjee, and S. Pendurkar. Semantic guided deep unsupervised image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11752 LNCS: 499–510, 2019. ISSN 16113349. doi: 10.1007/978-3-030-30645-8_46. 22, 33

S. Saha, L. Mou, C. Qiu, X. X. Zhu, F. Bovolo, and L. Bruzzone. Unsupervised Deep Joint Segmentation of Multitemporal High-Resolution Images. *IEEE Transactions on Geoscience and Remote Sensing*, 58(12):8780–8792, 2020. ISSN 15580644. doi: 10.1109/TGRS.2020.2990640. 23, 33, 34

I. H. Sarker. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(3), 2021. ISSN 2662-995X. doi: 10.1007/s42979-021-00592-x. 25

H. S. Seong, W. Moon, S. Lee, and J.-P. Heo. Leveraging hidden positives for unsupervised semantic segmentation, 2023. 68

E. Shelhamer, J. Long, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4): 640–651, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2572683. 13, 18

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. ISSN 01628828. doi: 10.1109/34.868688. 20

Z. Shi, Y. Yang, T. M. Hospedales, and T. Xiang. Weakly-Supervised Image Annotation and Segmentation with Objects and Attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2525–2538, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2645157. 27

W. Shimoda and K. Yanai. Distinct class-specific saliency maps for weakly supervised semantic segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9908 LNCS:218–234, 2016. ISSN 16113349. doi: 10.1007/978-3-319-46493-0_14. 27

C. Shorten and T. M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1), 2019. ISSN 21961115. doi: 10.1186/s40537-019-0197-0. 32

Simonyan Karen and Zisserman Andrew. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, page 14, 2015. URL `https://www.researchgate.net/publication/265385906_Very_`

Deep_Convolutional_Networks_for_Large-Scale_Image_Recognition%0Ahttp://www.robots.ox.ac.uk/. 18

B. Sirmacek and C. Unsalan. A probabilistic framework to detect buildings in aerial and satellite images. *IEEE Transactions on Geoscience and Remote Sensing*, 49(1):211–221, 2011. ISSN 01962892. doi: 10.1109/TGRS.2010.2053713. 43

N. Souly, C. Spampinato, and M. Shah. Semi Supervised Semantic Segmentation Using Generative Adversarial Network. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-October:5689–5697, 2017. ISSN 15505499. doi: 10.1109/ICCV.2017.606. 19

R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 2576–2582, 2017. 27

M. Tang, A. Djelouah, F. Perazzi, Y. Boykov, and C. Schroers. Normalized Cut Loss for Weakly-Supervised CNN Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1818–1827, 2018a. ISSN 10636919. doi: 10.1109/CVPR.2018.00195. 27

M. Tang, F. Perazzi, A. Djelouah, I. B. Ayed, C. Schroers, and Y. Boykov. On Regularized Losses for Weakly-supervised CNN Segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11220 LNCS:524–540, 2018b. ISSN 16113349. doi: 10.1007/978-3-030-01270-0_31. 27

M. Thoma. A survey of semantic segmentation. *CoRR*, abs/1602.06541, 2016. URL http://arxiv.org/abs/1602.06541. 13, 14

J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3001–3008, 2013. ISSN 10636919. doi: 10.1109/CVPR.2013.386. 27

Y. VERMA. A Complete Guide to Weak Supervision in Machine Learning, 2021. URL https://analyticsindiamag.com/a-complete-guide-to-weak-supervision-in-machine-learning/. 27

J. Wang, C. H. Ding, S. Chen, C. He, and B. Luo. Semi-supervised remote sensing image semantic segmentation via consistency regularization and average update of pseudo-label. *Remote Sensing*, 12(21):1–16, 2020a. ISSN 20724292. doi: 10.3390/rs12213603. 28

R. Wang, T. Lei, R. Cui, B. Zhang, H. Meng, and A. K. Nandi. Medical image segmentation using deep learning: A survey. *IET Image Processing*, 16(5):1243–1267, 2022. ISSN 17519667. doi: 10.1049/ipr2.12419. 12

S. Wang, W. Chen, S. M. Xie, G. Azzari, and D. B. Lobell. Weakly supervised deep learning for segmentation of remote sensing imagery. *Remote Sensing*, 12(2), 2020b. ISSN 20724292. doi: 10.3390/rs12020207. 27

Wikipedia. Weak Supervision, 2021. URL https://en.wikipedia.org/wiki/Weak_supervision. 27

X. Xia and B. Kulis. W-net: A deep model for fully unsupervised image segmentation. *CoRR*, abs/1711.08506, 2017. URL http://arxiv.org/abs/1711.08506. 20, 21, 31, 48, 49, 50

X. Yuan, J. Shi, and L. Gu. A review of deep learning methods for semantic segmentation of remote sensing imagery. *Expert Systems with Applications*, 169, 2021. ISSN 09574174. doi: 10.1016/j.eswa.2020.114417. 12

D. Zhang, C. Li, H. Li, W. Huang, L. Huang, and J. Zhang. Rethinking Alignment and Uniformity in Unsupervised Image Semantic Segmentation, 2022. URL http://arxiv.org/abs/2211.14513. 68

M. Zhang, Y. Zhou, J. Zhao, Y. Man, B. Liu, and R. Yao. A survey of semi- and weakly supervised semantic segmentation of images. *Artificial Intelligence Review*, 53 (6):4259–4288, 2020. ISSN 15737462. doi: 10.1007/s10462-019-09792-7. 14

Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. *Advances in Neural Information Processing Systems*, 2(January):1260–1268, 2014. ISSN 10495258. 27

L. Zhou and W. Wei. DIC: Deep Image Clustering for Unsupervised Image Segmentation. *IEEE Access*, 8:34481–34491, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.2974496. 7, 24, 33, 48, 49, 50, 51, 57, 58, 59, 67

Z. H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018. ISSN 2053714X. doi: 10.1093/nsr/nwx106. 26, 27

J. Zhu, J. Mao, and A. Yuille. Learning from weakly supervised data by the expectation loss SVM (e-SVM) algorithm. *Advances in Neural Information Processing Systems*, 2 (January):1125–1133, 2014. ISSN 10495258. 27

Appendix A

Ablation Studies

In this chapter, we perform additional ablation studies in our unsupervised semantic segmentation pipeline to further test and justify the choice of specific components or hyperparameter values.

A.1 Learning Rate

The first experiment we performed was regarding the learning rate (lr) we would use to train our proposed network in future experiments. This is important to check whether our chosen feature extractors could learn properly during training in the remaining experiments. We decided to use an exponential decaying learning rate from the beginning of the training until the end so that the network could perform smaller steps toward the gradient as the clustering process reached convergence. Three experiments were performed:

- 1 - $lr = 1e^{-3}$ to $lr = 1e^{-4}$
- 2 - $lr = 1e^{-4}$ to $lr = 1e^{-5}$
- 3 - $lr = 1e^{-5}$ to $lr = 1e^{-6}$

We start by checking the loss curves for each model training. As explained in Section 5.1.1, we do not plot any supervised metrics curve since we are not using a cross-validation set. Also, since this is an unsupervised problem, in practice, we would not have any supervision to compare; therefore, we only evaluate the loss curve of the training set in this parameter optimization experiment.

An example of the loss curves generated during training can be seen in Figure 36. Since our batch size equals 10 and the dataset has 7550 images, each epoch will have 755 batches. And since we train for 10 epochs, the total number of batches for each training is 7550. In this figure, as can be seen in the upper plots, the actual losses returned from the network contain a high amount of noise, which is caused both because the training is unsupervised, therefore subject to noise, and also because of the weighted cross entropy which for every batch assigns distinct weights to different classes. To alleviate this noise in the visualization, we use a moving average filter with a window size of 100 observations, resulting in the bottom plots, which will be the default visualization for the remainder of this work.

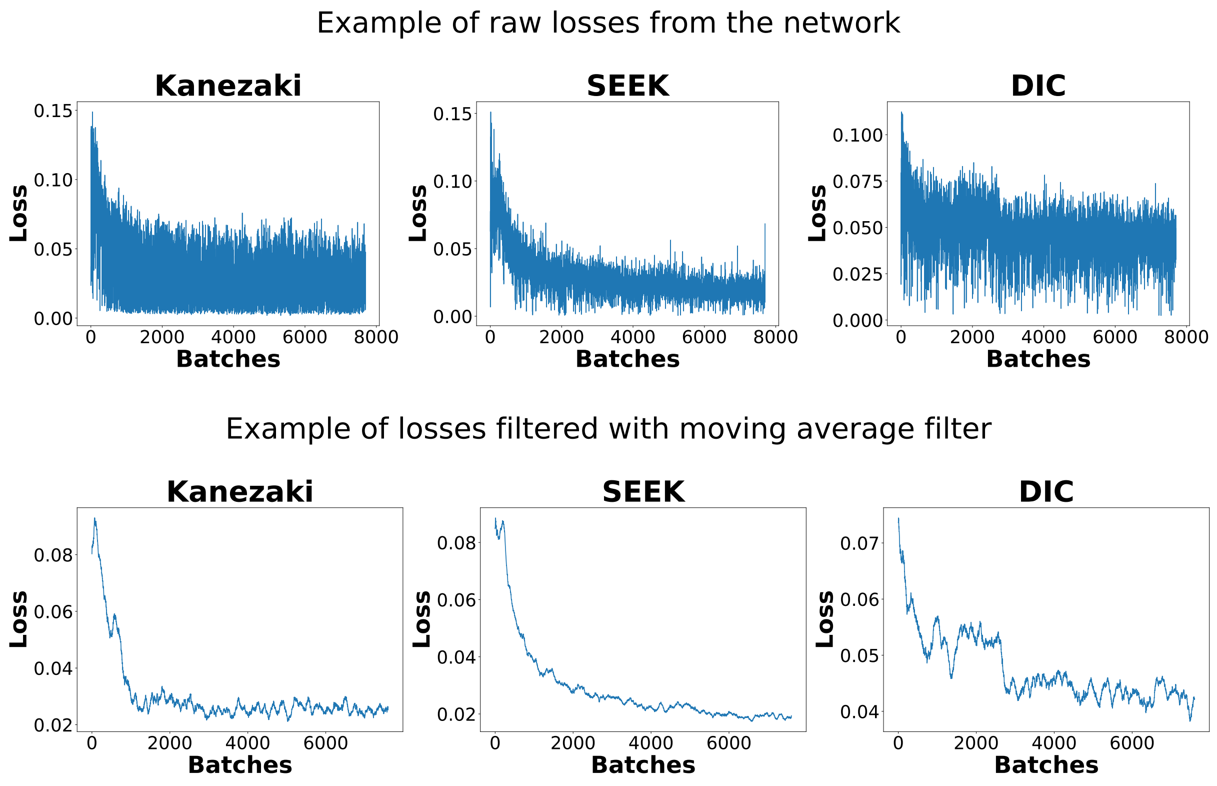


Figure 36 – Example of losses from an average training.

Source: produced by the author.

The loss curves for each experiment are aggregated in Figure 37, where the results for experiment 1 are replicated for convenience.

From the loss curves, we can see that both Kanezaki and SEEK converged fast in experiment 1, indicating that the learning rate might be too high. As for experiment 3, the losses suggest that no algorithm is learning correctly. Therefore, although the algorithms might not have entirely achieved convergence in experiment 2 due to its smoothness, this experiment seems to have achieved the best result.

We also evaluate our networks' metrics by dividing the experiments for each feature extractor. The results for the test dataset can be seen in Table 13, where "Exp" stands for the experiment number, "FE" stands for "Feature Extractor," and the remaining columns are the abbreviation of each metric, as described in Section 4.1. The "Kanezaki" feature extractor was also abbreviated for "Kan." For every feature extraction, there are

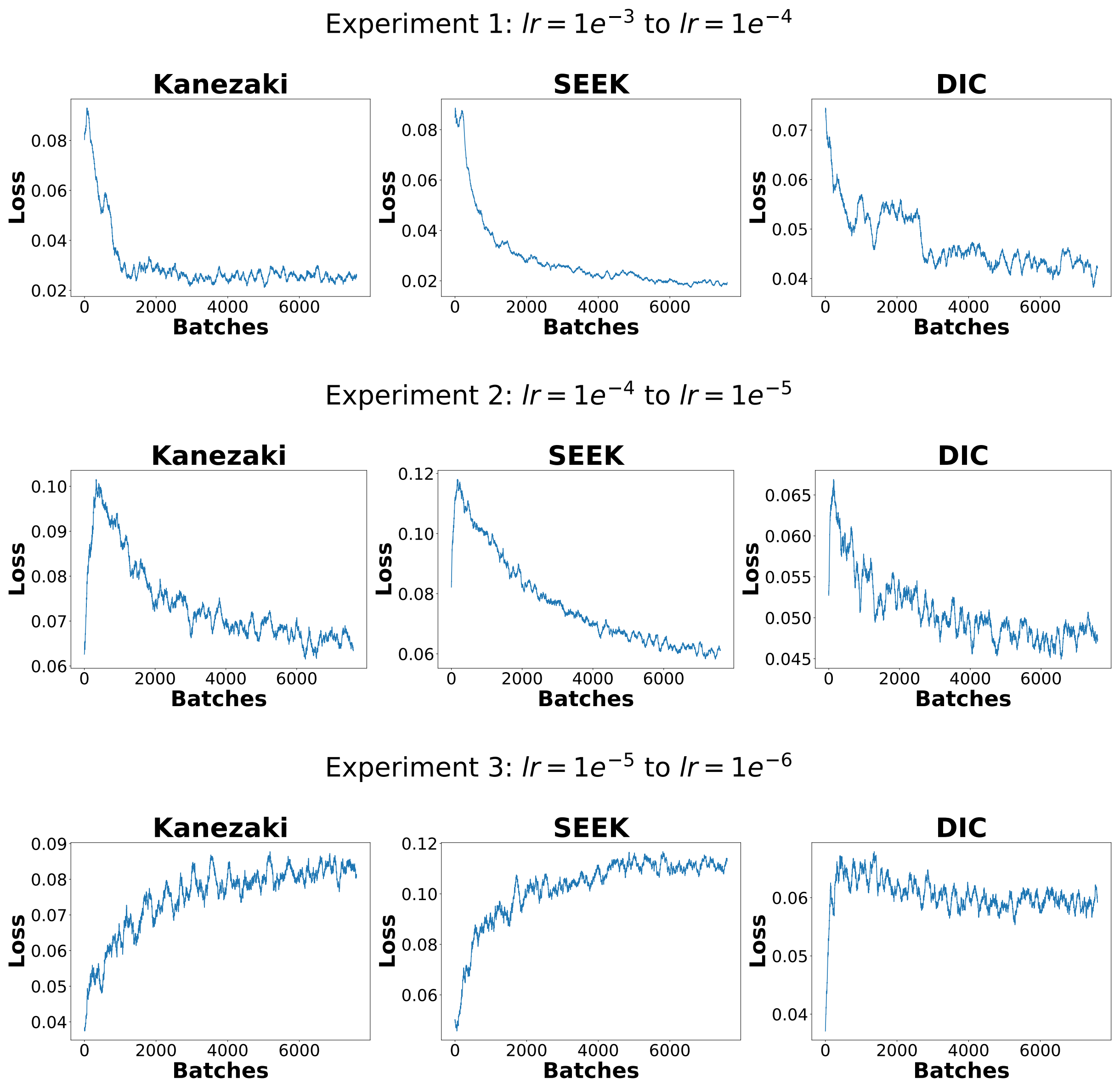


Figure 37 – Losses for the learning rate experiment.

Source: produced by the author.

two results, one for the complete dataset with all six classes and another where the name of the feature extractor has a "3", meaning that it is the variation of the dataset with only three classes. In this table, the IoU and wIoU metrics are the mean over all the classes, and the segmentation quantitative metrics (VoI, PRI, GCE, BDE, and SC) are all the mean of the metric value for every image in the test dataset. We also separate the types of metrics with a thicker vertical line between $F1$ and VoI, where the left half represents the supervised metrics, and the right half represents the segmentation metrics.

As the results show, experiment 2 obtained the best results overall. In experiment 3, DIC got the best Acc for the six classes experiment, but its IoU, wIoU, and F1-Score were better in experiment 2. Checking the other metrics, we can see that VoI, PRI, and

Table 13 – Metrics for the learning rate experiment.

Exp	FE	Acc↑	IoU↑	wIoU↑	F1↑	VoI↓	PRI↑	GCE↓	BDE↓	SC↑
1	SEEK	47.9	18.6	28.4	27.3	**1.50**	0.634	**0.167**	18.0	0.504
	DIC	59.3	28.5	41.8	39.4	1.65	**0.703**	0.260	11.8	0.538
	Kan	57.6	28.6	39.0	40.6	1.70	0.691	0.254	11.9	0.547
2	SEEK	58.6	24.6	38.6	33.0	1.58	0.678	0.217	12.6	**0.567**
	DIC	59.4	**32.4**	**43.2**	**45.7**	1.84	0.698	0.293	**11.1**	0.548
	Kan	58.4	26.8	40.0	37.3	1.83	0.686	0.293	11.2	0.535
3	SEEK	54.6	22.0	34.4	30.5	1.68	0.663	0.239	12.6	0.544
	DIC	**60.0**	30.2	42.2	42.3	1.76	0.698	0.274	11.8	0.558
	Kan	57.3	23.7	37.2	32.2	1.76	0.676	0.273	11.4	0.537
1	SEEK3	63.3	44.1	45.3	60.2	1.16	0.704	**0.159**	20.2	0.654
	DIC3	72.5	55.4	56.3	70.8	**1.15**	**0.748**	0.173	15.7	0.677
	Kan3	71.6	54.7	55.8	70.0	1.35	0.724	0.214	14.7	0.664
2	SEEK3	72.8	56.4	57.4	71.6	1.27	0.733	0.202	14.5	**0.688**
	DIC3	**74.1**	**58.4**	**59.4**	73.2	1.27	0.740	0.203	**13.8**	0.679
	Kan3	72.7	56.4	57.2	71.7	1.40	0.717	0.225	14.8	0.667
3	SEEK3	69.2	51.5	52.6	67.1	1.35	0.714	0.210	15.2	0.669
	DIC3	73.9	58.3	59.2	**73.3**	1.32	0.725	0.211	15.0	0.681
	Kan3	71.3	54.7	55.6	70.3	1.48	0.700	0.235	15.0	0.657

GCE are better for experiment 1 in general, while GCE and SC are better for experiment 2.

Given these results, we chose to follow the experiments with our learning rate decaying from $1e^{-4}$ to $1e^{-5}$ since it obtained the more consistent supervised metrics for every feature extractor, competitive segmentation metrics compared to the other experiments, and it also presented a smooth loss function curve.

A.2 Number of classes

In our proposed architectures, we used several classes, or clusters, of $K = 100$ since having a large number of classes should theoretically allow the network to learn a wider variety of semantics. But we also tested this hypothesis by training our network with a lower number of classes, $K = 10$, and compared it with the results from Section A.1 (which already is the result for $K = 100$). Here we used only the best learning rate achieved in the previous experiment. The loss curves can be seen in Figure 38.

Clustering images with a lot of content variation, which is the case for remote sensing images, is a challenging problem. The loss functions show that the network had more difficulty learning to segment using only 10 classes since the loss values are higher and the curves are less steep. The curves show that a higher number of classes allow the network to distribute the semantics between more classes and learn more specific semantic features. Next, in Table 14, we evaluate the metrics for this experiment, where "K" is the

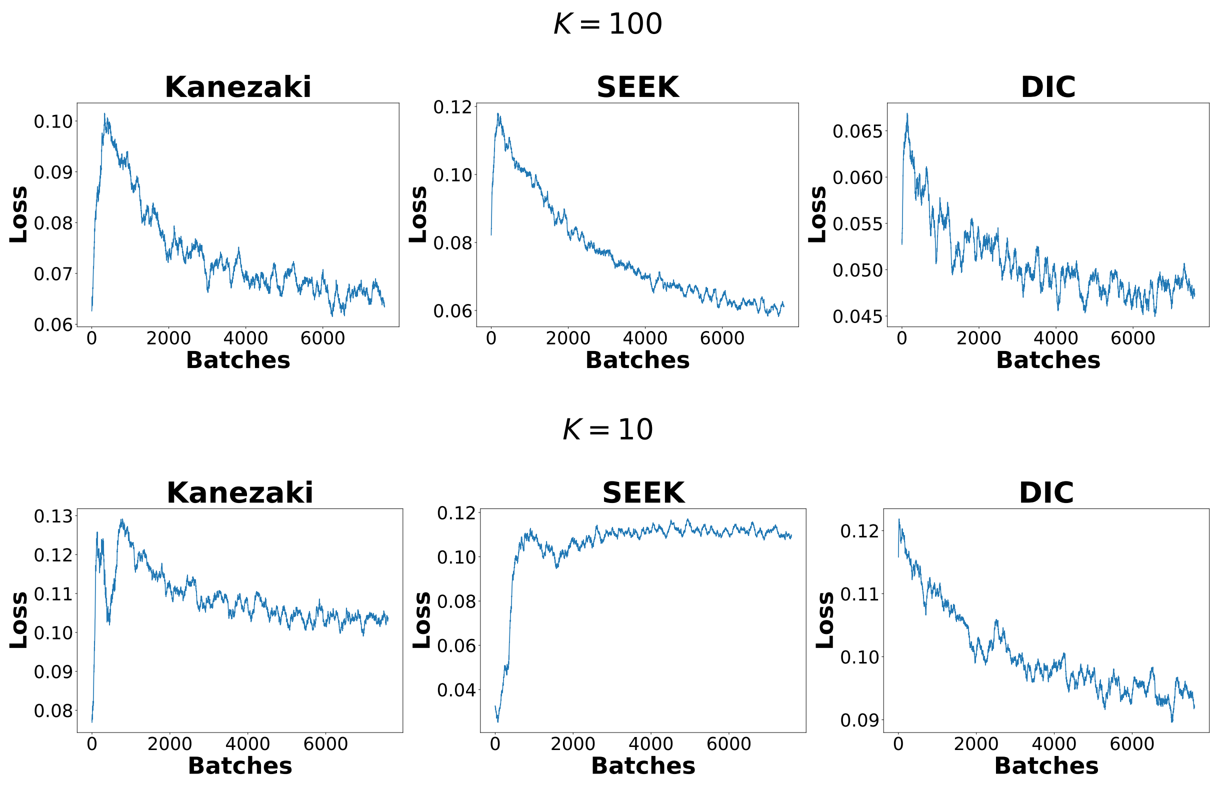


Figure 38 – Losses for the class number experiment.

Source: produced by the author.

number of classes.

Table 14 – Metrics for the class number experiment.

K	FE	Acc↑	IoU↑	wIoU↑	F1↑	VoI↓	PRI↑	GCE↓	BDE↓	SC↑
100	SEEK	58.6	24.6	38.6	33.0	1.58	0.678	0.217	12.6	0.567
	DIC	59.4	**32.4**	**43.2**	**45.7**	1.84	0.698	0.293	**11.1**	0.548
	Kan	58.4	26.8	40.0	37.3	1.83	0.686	0.293	11.2	0.535
10	SEEK	50.2	19.5	30.2	28.0	1.70	0.652	0.243	13.0	0.545
	DIC	**59.9**	25.6	40.1	33.8	**1.41**	**0.709**	**0.186**	14.1	**0.585**
	Kan	54.4	21.9	34.3	30.4	1.72	0.659	0.254	12.9	0.549
100	SEEK3	72.8	56.4	57.4	71.6	1.27	0.733	0.202	14.5	0.688
	DIC3	74.1	**58.4**	**59.4**	**73.2**	1.27	**0.740**	0.203	**13.8**	0.679
	Kan3	72.7	56.4	57.2	71.7	1.40	0.717	0.225	14.8	0.667
10	SEEK3	65.8	47.6	48.9	63.4	1.38	0.702	0.213	15.3	0.652
	DIC3	**74.5**	58.1	59.1	73.0	**1.12**	0.765	**0.173**	14.9	**0.703**
	Kan3	68.6	51.4	52.4	67.3	1.42	0.699	0.219	15.4	0.659

Overall, many segmentation quality metrics are better for the 10 classes problem. This is indicative that even though the classes learn more specific semantics in the case with more classes, when mapping this information to the particular classes of the dataset, part of the information is lost, meaning that the usage of the classes without the mapping might make more sense.

For the supervised metrics on the left side of the table, the 10 classes experiment obtained a better Acc. Still, the IoU, wIoU, and F1-score all were better for the 100 classes network, which is more important in a semantic segmentation problem than just the accuracy. With these results, we choose to stay with 100 classes in our network.

A.3 Cross-Entropy

The third experiment that we performed was regarding cross-entropy loss. We proposed using a weighted version of this loss, where the weights are recalculated at every batch. But this causes the loss function to become very unstable between batches, as seen in Figure 36. To explore this problem, we trained our network with our best learning rate and many classes, varying between our loss function and the original cross-entropy, and compared the results. The loss function plots can be seen in Figure 39, where "LF" stands for "Loss Function."

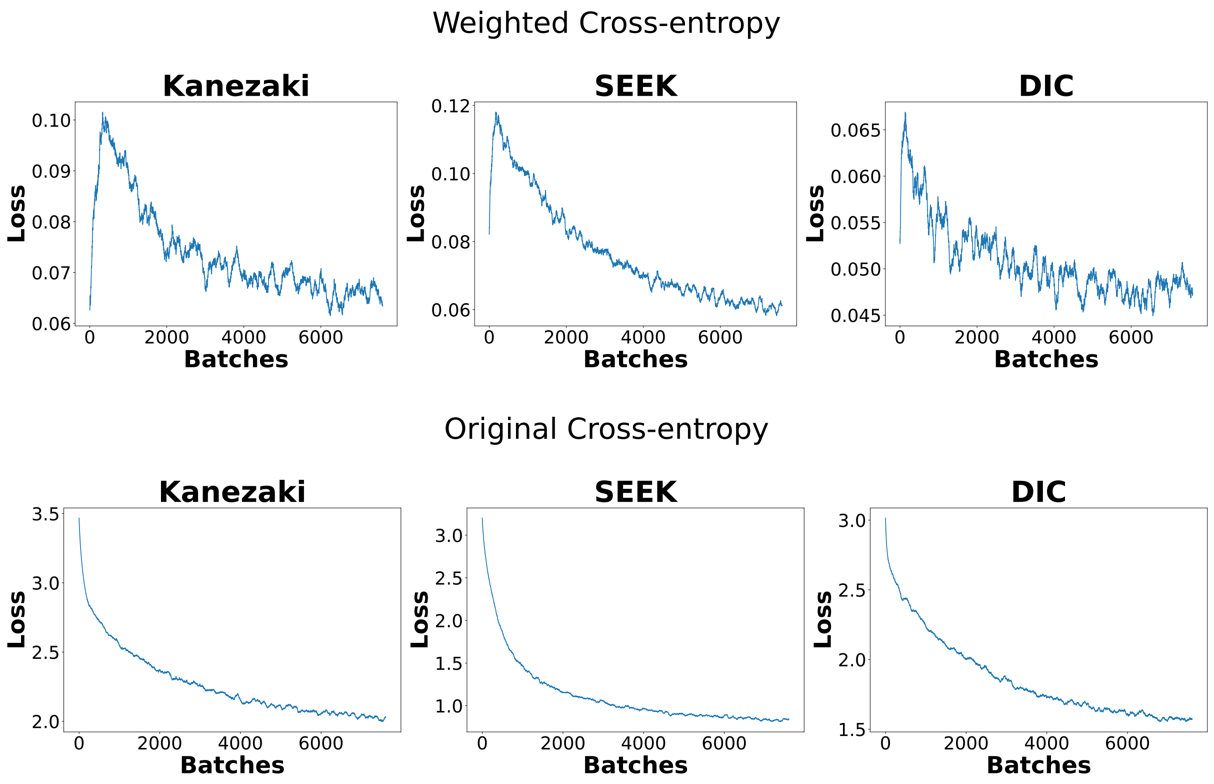


Figure 39 – Losses for the cross-entropy experiment.

Source: produced by the author.

As we can see, both networks trained similarly, but the original cross-entropy loss was smoother than our proposed loss. Here the values cannot be compared since they are in different scales. Next, in Table 15, we evaluate the metrics for each loss.

Table 15 – Metrics for the cross-entropy experiment.

LF	FE	Acc↑	IoU↑	wIoU↑	F1↑	VoI↓	PRI↑	GCE↓	BDE↓	SC↑
WCE	SEEK	58.6	24.6	38.6	33.0	**1.58**	0.678	**0.217**	12.6	0.567
	DIC	59.4	**32.4**	**43.2**	**45.7**	1.84	0.698	0.293	**11.1**	0.548
	Kan	58.4	26.8	40.0	37.3	1.83	0.686	0.293	11.2	0.535
CE	SEEK	50.3	19.3	29.7	27.5	1.67	0.658	0.234	12.9	0.543
	DIC	**59.5**	27.9	41.3	38.7	1.61	**0.708**	0.254	11.8	0.549
	Kan	57.2	23.8	37.3	32.2	**1.58**	0.683	0.223	12.8	**0.579**
WCE	SEEK3	72.8	56.4	57.4	71.6	1.27	0.733	0.202	14.5	0.688
	DIC3	**74.1**	**58.4**	**59.4**	**73.2**	1.27	0.740	0.203	**13.8**	0.679
	Kan3	72.7	56.4	57.2	71.7	1.40	0.717	0.225	14.8	0.667
CE	SEEK3	65.1	45.4	46.7	60.1	**1.15**	0.736	**0.169**	14.6	0.692
	DIC3	72.9	56.5	57.5	71.5	1.17	**0.752**	0.181	14.6	**0.700**
	Kan3	71.5	54.4	55.3	69.9	1.29	0.731	0.202	15.2	0.683

The supervised metrics on the left side of the table show that the only case where the CE performed better was the accuracy of the six classes dataset, by 0.1%. The remaining metrics were all the better for the WCE loss. As for the segmentation metrics on the right side of the table, results vary between methods. The CE method obtained a slightly better SC value in both experiments, indicating that some regions overlap better for this loss, but this could be the case because since there are no class weights, the larger classes might be favored, and if these contain the largest homogeneous regions in the dataset images, it makes sense that the metric evaluating the region overlap would have a better value. For the remaining segmentation metrics, most obtained a better result for the CE loss in the three classes dataset. Since the classes are better balanced in this case, the impact of favoring smaller classes is lower, explaining why the segmentation metrics are better. Still, the supervised metrics indicate that the WCE loss performed better.

In general, this experiment proves that using a class weighted loss approach can favor the smaller classes, and since the training is unsupervised, balancing the weights is an essential way of regularizing the learning and avoiding that larger classes overfit the network by assigning every pixel to them, mainly at the beginning of the training.

With this experiment, we also settle for the DIC feature extractor. Even though, in practice, we kept testing the other feature extractors for the remaining experiments in the Results chapter, at this point, there is enough information, when comparing all of the previous experiments' metrics tables, that the DIC feature extractor has the best overall results and can generalize better than the other classifiers, probably because of both the more extensive network and the DCS block.