UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Pedro Henrique Araujo Pinto

Semantic Segmentation with Siamese Autoencoder and Latent Data Model via Context Windows

Belo Horizonte
2022

Pedro Henrique Araujo Pinto

Semantic Segmentation with Siamese Autoencoder and Latent Data Model via Context Windows

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Jefersson Alex dos Santos

Belo Horizonte
2022

Pinto, Pedro Henrique Araújo

P659s Semantic segmentation with siamese autoencoder and latent data model via context windows [manuscrito] / Pedro Henrique Araújo Pinto — 2022.
80 f. il.; 29 cm.

Orientador: Jefersson Alex dos Santos.
Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação
Referências: f. 76-80

1. Computação – Teses. 2. Redes neurais (Computação) – Teses. 3. Segmentação semântica – Teses. 4. Sensoriamento remoto – Teses. I Santos, Jefersson Alex dos. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. III. Título.

CDU 519.6*82 (043)

Ficha Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa CRB 6/1510 Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

SEMANTIC SEGMENTATION WITH SIAMESE AUTOENCODER AND LATENT DATA MODEL VIA CONTEXT WINDOWS

PEDRO HENRIQUE ARAÚJO PINTO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores(a):

Prof. Jefersson Alex dos Santos - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Heitor Soares Ramos Filho
Departamento de Ciência da Computação - UFMG

Prof. Cristiano Leite de Castro
Departamento de Engenharia Elétrica - UFMG

Belo Horizonte, 25 de julho de 2022.

Documento assinado eletronicamente por **Jefersson Alex dos Santos**, **Professor do Magistério Superior**, em 01/09/2022, às 06:28, 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 **Heitor Soares Ramos Filho**, **Professor do Magistério Superior**, em 02/09/2022, às 11:25, 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 **Cristiano Leite de Castro**, **Professor do Magistério Superior**, em 09/09/2022, às 17:15, 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 **1726050** e o código CRC **F14C0636**.

A Amantino da Silva Marreco.

Nam quis esset, quid fecit, et quid reliquit. Vide te mox.

Acknowledgments

I thank my parents for believing, investing, and supporting me on this rewarding journey. There are not enough words to thank you for such love and unconditional support, even with the distance between us. It is a privilege to have been raised, influenced and guided by such admirable, sincere and upright people. This achievement is also for you, we will celebrate together.

To Professor Jefersson Alex dos Santos, my infinite thanks for the guidance, support, assistance and understanding for the accomplishment of this work. There is no question to be asked about such a human, helpful and supportive person. Oh if all teachers were like you...

To my best friend, Henrique Cançado. A companion and brother that life gave me when looking for a new home in Belo Horizonte. Every year growing together, cherishing the collective. The daily learnings of our relationship surpass any scientific knowledge possible to acquire. You can always count on me. To my love, Carolina Alzamora, for her patience during this arduous journey and for staying by my side, supporting me, during the most intimate frustrations. Life with you is better than I ever imagined. I love and admire you.

To all my colleagues, members or supporters of Let It Trip, for providing me with welcome and escape valves for all moments of anxiety. Our bond is very strong and eternal. Finally, I thank myself for the nights of sleep lost in the midst of hard work, for the resilience to give up leisure activities in favor of a greater goal and for the commitment to a challenging personal project. Worth it.

"*Thus, the task is not so much to see what no one yet has seen, but to think what nobody yet has thought about that which everybody sees.*"

(Arthur Schopenhauer)

Resumo

Sensoriamento remoto é o conjunto de técnicas e procedimentos tecnológicos que visa à representação da superfície terrestre sem a necessidade de um contato direto e envolve ações para levantar dados, informações e imagens da superfície, com o intuito de representá-las e melhor entender os seus aspectos. Com o avanço tecnológico e consequente aumento de dados obtidos para análise, juntamento com o aprimoramento de técnicas de redes neurais artificais cada vez mais poderosas, diversas tarefas de visão computacional - como segmentação semântica - têm atraído cada vez mais atenção de pesquisadores.

Segmentar uma imagem aérea de alta dimensão, apesar de não ser uma tarefa fácil, tem apresentado resultados promissores com o uso de redes neurais. Diversas variações de arquiteturas e módulos de auxílio - como módulos de atenção - para classificação de pixels foram testados na literatura para segmentação de imagens. No entanto, a segmentação de imagens aéreas ainda apresenta espaço para melhora e algumas frentes de trabalho pouco exploradas.

Nesse trabalho, utilizamos o aprendizado métrico profundo para a segmentação de imagens aéreas em quatro cenários: prédios (construções), plantações de café, carros e árvores. Utilizamos uma arquitetura, chamada SMELL, originalmente desenvolvida para tarefas de classificação e a adaptamos para solucionar problemas de segmentação semântica utilizando janelas de contexto. A aplicação de uma rede neural siamesa, com um módulo de aprendizado métrico para o qual a função de distância é aprendida e optimizada pelo próprio modelo parece não ter sido explorada na literatura para sensoriamento remoto.

Nossos testes mostram que a utilização de distâncias para a classificação a nível de pixel pode ser muito útil para tarefas de segmentação, superando algumas arquiteturas que figuram o estado da arte, como ResNet e Xception. Nosso trabalho abre espaço para a exploração de outras técnicas de aprendizado métrico, bem como apresenta possíveis melhorias a serem testadas no método apresentado.

Palavras-chave: redes neurais artificiais, segmentação semântica, sensoriamento remoto.

Abstract

Remote sensing is the set of techniques and technological procedures that aim to represent the earth's surface without the need for direct contact and involves actions to collect data, information and images from the surface, in order to represent them and better understand their aspects. With technological advances and the consequent increase in data obtained for analysis, together with the improvement of increasingly powerful artificial neural network techniques, several computer vision tasks - such as semantic segmentation - have attracted increasing attention from researchers.

Segmenting a high-dimensional aerial image, despite not being an easy task, has shown promising results with the use of neural networks. Several variations of architectures and aid modules - such as attention modules - for pixel classification were tested in the literature for image segmentation. However, the segmentation of aerial images still has room for improvement and some work fronts little explored.

In this work, we used deep metric learning to segment aerial images in four scenarios: buildings (constructions), coffee plantations, cars and trees. We used an architecture, called SMELL, based on an autoencoder and a distance learning module, originally developed for classification tasks and adapted it to solve semantic segmentation problems using context windows. The application of a siamese neural network, with a metric learning module for which the distance function is learned and optimized by the model itself, seems not to have been explored in the literature for remote sensing.

Our tests show that the use of distances for classification at the pixel level can be very useful for segmentation tasks, surpassing some state-of-the-art architectures, such as ResNet and Xception. Our work opens space for the exploration of other metric learning techniques, as well as presents possible improvements to be tested in the presented method.

Keywords: artificial neural networks, semantic segmentation, remote sensing.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Images are a two dimensional representation of a scene that resembles a subject — usually a physical object — and thus provides a depiction of it. It can also be interpreted as a distributed amplitude of colors that generates context through visual perception [29]. The amount of detail in an image is directly related to its pixel count: the more pixels, the more details, the more resolution.

With the increase in the accuracy and speed of automated processes for extracting, adding or manipulating information from images, its applications have expanded rapidly, adapting computing techniques to solve the most diverse types of tasks. The study of computer algorithms that can automatically improve through experience and through the use of data is called Machine Learning and groups together theories of statistical modeling and computer science [28]. Concurrently, the field of computer science that applies Machine Learning techniques and seeks to understand, replicate and automate tasks that the human visual system can do is called computer vision.

In current usage, the term remote sensing generally refers to the use of satellite or aircraft-based sensor technologies to obtain high-resolution aerial images of the Earth's surface. Improvements in such technologies have significantly increased the availability of high spatial resolution images and the diffusion of the study of their numerous applications, such as classification, segmentation and detection became inevitable.

Semantic segmentation, also known as pixel-wise classification, is one of many computer vision tasks in which Neural Networks are widely applied and consists of labeling each pixel of an image with a corresponding class of what is being represented. It is a form of pixel-level forecasting because each pixel in an image is classified according to a category. Although the task consists of categorizing each pixel, taking into account the surrounding context is very useful, as the pixels of an image represent the spatial relation between objects. Due to the increase of performance by capturing neighborhood context and the capacity of learn specific adaptable features while, at the same time, learn classifiers, convolutional neural networks (CNNs) have been widely applied to semantic segmentation tasks.

Despite that, CNNs present a technical problem regarding contextual information, due to convolution operations with local receptive fields. As pointed out by [40, 15],

this contextual information of an image is capable of providing important characteristics for pixel-wise classification. Moreover, since each neuron of a CNN models a specific input region, architectures with only convolutional layers fails in modeling global spacial relations in an effective way [30].

To better model global spacial relations, [31] proposed a novel technique to automatically perform pixel-wise land cover classification based on data-driven feature descriptors for high-resolution remote sensing images, using CNNs. The proposed land-cover mapping method partitions one image for each pixel with a context size - smaller compared to the entire image - and assign it a label equals to the class that the center pixel of this image belongs to. After training the CNN the image's mask is reconstructed using the predicted category of each image as one pixel.

As in this case, the CNNs architectures have proved to be versatile for the most diverse tasks in computer vision. Inevitably, several adaptations emerged, each with its purpose. Although many variations of neural networks has been adapted to deal with semantic segmentation problems, Metric Learning based approaches seems little explored in remote sensing, being more commonly found for medical images datasets [19, 23].

Metric Learning aims to learn data embeddings/feature vectors in a way that reduces the distance between feature vectors corresponding to the same category and increases the distance between the feature vectors corresponding to different labels. Different Machine Learning problems frequently require a distinct notion of similarity semantics, which is typically poorly captured by standard distance metrics. In order to learn distance metrics from the input data, Siamese Neural Networks (SNNs) are typically used [7].

[4] propose a new metric learning method - named SMELL - defined in a new latent space obtained through a siamese autoencoder for image classification. In their work, the power of metric learning is evidenced by the overcoming of 13 different distance metric learning approaches with SMELL.

Here, we propose the adaptation of this deep metric learning method - originally designed for image classification - for binary semantic segmentation of aerial images, using [31] context windows method to adjust the training process for SMELL. The experimental setup can be summarized as follow:

1. We test different context windows sizes for multiple datasets aiming to evaluate the performance gain due to more or less contextual information. In the original paper of the context windows methodology, [31] shows that different quantities of contextual information are better suited for different scenarios, testing 7×7 and 25×25 windows size. Here, we explore a more continuous and better defined space, adapted to each scenario.

2. We measured SMELL's performance against three state-of-art neural networks when

trained using context windows methodology: ResNet50, ResNet101 and Xception. These networks architectures were chosen for being easily implemented using Python's Keras applications framework and because they are adaptable architectures that serve as baseline for computer vision tasks and therefore represent the state of the art in recent years.

3. We evaluate the impact of the amount of contextual information used in four different binary segmentation scenarios: building, coffee plantation, tree and car. We used the Inria dataset for building, the Brazilian Coffee Scenes datasets for coffee plantations and the Potsdam for tree and car segmentation.

4. We tested SLIC superpixel algorithm aiming to enhance SMELL's performance for each scenario. We vary the number of superpixels and the compactness parameters over a wide range to find the most suitable configuration for each case.

The obtained results show that our proposal minimally matches state-of-art architectures performance in all cases. We applied SLIC superpixel algorithm as a post processing method to SMELL's predictions increasing even more the accuracy for the best context windows sizes. We present some visual representations of the outputs comparing to the ground truth masks, making it possible to conclude that the results generated by our proposal are more accurate and more like the objective ground truth mask.

1.1 Motivation

Processing and analyzing aerial images play central roles in terrain modeling, agricultural monitoring, city planning, environmental surveillance, etc. Aerial images are developing towards high resolution and large size, which poses a major challenge in pixel-level image segmentation. Regardless the rapid development of deep learning technology and the satisfactory level of precision already achieved, the versatility of possible applications keep this field of study on the rise among researchers. Furthermore, with the increasing accessibility to high spatial resolution images, it has become easier and cheaper to perform experiments with neural networks for high-resolution images, shortening the road to be traveled.

Learning distances, i.e. Metric Learning (MeL), from data is a common attempt to improve machine learning approaches. For the semantic segmentation of aerial images, however, its adaptation seems little explored while potentially rewarding. The statistical

background that underpins the application of metric learning and the infinity of possible ramifications resulting from this work were the biggest motivating factors for this one.

Lastly, the definition of a latent space - obtained through a siamese autoencoder - in which the pixels of an image are separated in regions of similarity/dissimilarity and the distance metrics can be simultaneously learned along with a latent representation of the data and the similarity markers proved to be effective for image classification. Therefore, its adaptation to semantic segmentation with context windows potentiates achieving relevant results if compared to other state-of-art architectures trained with the same approach.

1.2 Objectives

This work explores the adaptation of SMELL [4], a siamese convolutional autoencoder with a metric learning module for the latent space S, to semantic segmentation of aerial images using the context windows methodology, as in [31]. We intend to demonstrate the viability of a novel framework inspired by the application of Deep Metric Learning (DMeL) algorithms for semantic segmentation that is comparable to state-of-art architectures. We tested this adaptation in four tasks: buildings, coffee crops, car and trees segmentation. We also verify the impact in accuracy due to the amount of contextual information that is provided to the autoencoder and the performance. We exploit a more continuous and well defined space than [31] for the building and coffee crops scenarios. For car and tree segmentation experiments, we tested one small context window size against a high degree of contextual information. We also explore the application of SLIC superpixel technique as post-processing method aiming to improve the final segmentation masks generated.

1.3 Thesis Roadmap

The remainder of this work is organized as follows. Chapter 2 presents a brief review of literature, analyzing methods regarding semantic segmentation of aerial images, metric learning methods that enhance neural networks for computer vision recognition (Section 2.2) in different scenarios (object, cameras, handwritten digit, etc). In Chapter

3, basic concepts for a better understanding of the proposed approach are explored. The proposed methodology is detailed in Chapter 4 and a vast number of experiments is reported in Chapter 5. Finally, we conclude in Chapter 6, presenting final remarks and future works.

Chapter 2

Literature Review

This work intercepts some different research areas. We adapt a Deep Metric Learning method, composed by a Siamese Neural Network with a Metric Learning module, to perform semantic segmentation of aerial images. In this chapter we discuss works regarding Deep Metric Learning and Semantic Segmentation. We briefly present the definitions and main applications of each section, followed by an in-depth review of remote sensing studies. Finally, we also present a section about Deep Metric Learning for Semantic Segmentation of Aerial Images, combining all the topics of this research together.

2.1 Deep Metric Learning

Metric learning is an approach based directly on a distance metric that aims to establish similarity or dissimilarity between images [22]. Deep Metric Learning on the other hand uses Neural Networks to automatically learn discriminative features from the images and then compute the metric. MeL algorithms often use a linear projection, to measure the similarity among samples while using an optimal distance metric for learning tasks, what limits its capacity of solving real problems with non-linear characteristics. While the kernel trick can be adopted to address this non-linearity problem, Deep Metric Learning (DMeL) helps capture non-linear feature structure more accurately by learning a non-linear transformation of the feature space using activation functions.

Metric learning's main goal is to learn a new metric that will minimize distances between samples of the same class while increasing distances between samples of different classes. While metric learning seeks to bring similar things closer together, it increases the gap between dissimilar objects. The ability of a metric learning method to separate similar from dissimilar images is related to how well a space - over which distances will be computed - represents the data. Therefore, learning the similarity metrics between arbitrary images is a fundamental problem for a variety of tasks, such as image retrieval [35], face recognition - or person re-identification - [33, 41], localization [18], video tracking

[5], classification [21] and semantic segmentation [6].

When encountered with such problems, it is possible to make use of a metric space to come up with solutions. According to [25], there are two main types of neural networks used in DMeL methods: Siamese networks and triplet networks. Since the goal is to measure similarity, at least two samples of data are needed. In a Siamese network for classification scenario, we take an input image and find out the encodings of that image, then, we take the same network without performing any updates on weights or biases and input a different image and again predict it's encodings. Then, we compare these two encodings to check whether there is a similarity between the two images. These two encodings act as a latent feature representation of the images.

For a triplet loss based network the idea remains, however using three input images instead of two. One image is selected as an anchor image and we take a negative image - an image with a different label - and a positive one - with the same category. We train the network with shared weights trying to minimize the dissimilarity between the anchor and the positive images and maximize the dissimilarity between anchor and negative images.

[7] presents a method for training a similarity metric from data for face verification purposes. The learning process minimizes a discriminative loss function that drives the similarity metric to be small for pairs of faces from the same person, and large for pairs from different persons. Following this research area, in [16], the authors exploited discriminative information from neighborhood relationships of samples to learn the mapping function, presenting an important DMeL method called Dimensionality Reduction by Learning an Invariant Mapping (DrLIM). Their method only needs neighborhood relationships between training samples to learn distance functions that are robust to nonlinear transformations of the input signals, generating mappings that are smooth and coherent in the output space.

Later, [42] proposed a DMeL method for person re-identification, which learned a similarity metric from image pixels directly with Siamese networks. In their work, the authors violate the parameters sharing constraint - usually applied to Siamese networks - based on the assumption that without parameters sharing, the network can deal with the view specific matching tasks more naturally, since they split each input image in three parts. This paper was the first one to apply deep learning in the person re-identification problem and also the first work to study the person re-identification problem in cross database setting.

As DMeL methods for computer vision tasks improved, their applications naturally diversified in the sense that more complex challenges became more precisely solvable. [39] summarize the different types of features and metric learning approaches from a label attributes perspective for person re-identification tasks. The authors conducted experiments on the features and metrics in different categories. For deep feature extraction, AlexNet, ResNet-50, VGG16 and InceptionV3 were tested as network architectures while

other four handcraft feature extraction techniques were also tested. In the Metric Learning side, Euclidean was the only distance metric applied, among some other supervised and unsupervised methods, as FDA, LFDA and PCA.

Although the overviews of feature extraction and metric learning methods form a solid foundation on the importance of DMeL methods for computer vision tasks and the proposed taxonomy serves as a guide for future research, the results show that Metric Learning has a significant impact on handcraft features but not on deep features. In addition, it was discovered that the space of the solution is different from the originally extracted feature space, since the results of some modern metric learning methods are lower than the Euclidean distance.

With a different and more interesting perspective, [4] propose obtaining the feature representation into a latent space S using an autoencoder for image classification tasks. The method, namely SMELL, simultaneously optimize a new data representation using an Siamese Neural Network (SNN) model and a similarity function that indicates the similarity of two objects in the S space. The authors point out that metrics defined without any prior knowledge about the data such as Euclidean and Manhattan distance fails to appropriately measure distances in some datasets, in agreement with [39].

To find an appropriate similarity measure, the Metric Learning algorithm of SMELL define markers that represent one of two possibilities: same label for the input images or different labels. The closer the vector of the features obtained in the S space from a positive marker the greater the probability that the elements of the paired input are similar to each other. The closer the vector of the features from a negative marker the greater the probability that the elements of the paired input are dissimilar. Another interesting point about SMELL is that it uses the Student's t-distribution with one degree of freedom as a kernel to measure the similarity between the feature maps and the markers. Also, the feature extraction module and the Metric Learning algorithm are simultaneously trained, which encourages the similarity metric to be adequate to the problem.

2.1.1 Deep Metric Learning for Remote Sensing

In Remote Sensing area, the main applications of DMeL methods are image retrieval [43], image classification [21] and change detection [34].

[14] first introduce Deep Structural Metric Learning (DSMeL) into the literature of remote sensing scene classification. The authors adapt the idea of [36] - applied to image retrieval - to deal with scene classification tasks. The proposal (named D-DSML) was to incorporate DSMeL into deep networks through a special structured loss, which

would make the network capable of capturing contextual information, thus solving a common problem in the classical DMeL methods of the time. While these traditional methods treated training samples in each training batch with the stochastic gradient descent-based learning method independently, they neglected the contextual (structural) information in the training samples. The authors proposed the use of $O(m^2)$ pairwise distances within each batch instead of O(m) pairwise distances allowing the network to fully interpret the image's structural information. In addition, their work present a diversity-promoting prior as regularization on DSMeL to encourage the learned parameter factors to be diversified, decreasing the redundancy caused by the similarity of the metric parameter factors. Experiments conducted on six real-world remote sensing scene data sets validate the effectiveness and the wide applicability of the proposed method for scene classification, showing its competitiveness on scene classification at that time.

Later, [13] improved the representational ability of the previous DSMeL model by considering not only the pairwise correlation between the training samples but also the class correlation of the class view, making the model take full advantage of the training batch. To take the class-wise penalization into consideration, their methodology defines the center points of the learned features in the training process to represent its label. Their work develops a supervised joint learning of the SoftMax loss and the center-based structured metric learning to maximize the inter-class variance and minimize the intra-class variance for remote sensing imagery. Experimental results over three datasets have shown that the center points can improve the model's performance and the learned features can be more discriminative. Compared to D-DSMeL [14], the improvement in the accuracy was very low. The proposed method achieve a 97.30% mean accuracy for the UC Merced Land-Use Dataset, against 96.76% for the D-DSML model. For the Google Dataset, the D-DSML accuracy was almost the same (0.10% smaller).

Still in the image classification of high-resolution aerial images scenario, [8] propose a deep metric learning-based feature embedding model, which can meet the tasks both for same- and cross-scene Hyperspectral imagery classification. For the same-scene classification part, the authors use a Similarity Based Deep Metric Learning module to measure the metric value between two embedding features. In other words, their metric module is a neural network, which produces a metric value representing the similarity between any two embeddings. The higher the metric value, the greater the similarity. For the cross-scene challenge, an Unsupervised Domain Adaptation methodology is applied. The method's outstanding results in both the same- and cross-scene classifications corroborates as a demonstration of the power of Deep Metric learning for computer vision tasks by overcoming the state-of-art methods.

More recently, [21] focused on investigating the performance of the learned feature embeddings and the associated metric space for remote sensing scenes. Since most of the existing DMeL use a contrastive or a triplet loss, the authors argue that the straight-

forward application of these techniques to remote sensing images may not be optimal in order to capture their neighborhood structures in the metric space due to the insufficient sampling of image pairs or triplets during the training stage and to the inherent semantic complexity of remotely sensed data. To overcome this limitation for the remote sensing imagery scenario, they propose a new DMeL approach with two components: a scalable neighborhood component analysis (SNCA) and the cross-entropy loss. While the first component aims at discovering the neighborhood structure in the metric space, the second one aims at preserving the class discrimination capability based on the learned class prototypes. By testing in two different datasets with different perspectives - classification, clustering and image retrieval - the results showed that the SNCA moduel with a cross entropy loss can outperform state-of-art architectures in all perspectives.

Although in the last couple of years there has been a lot of architectural variations of DMeL methods to tackle different problems, such as remote sensing image retrieval with a cost-sensitive loss and intraclass space sample mining [45] and rotation invariant methods for remote sensing image classification [20], the applications of DMeL methods for semantic segmentation of remote sensing imagery still seems to be poorly explored in literature.

2.2 Semantic Segmentation

Image segmentation is a computer vision task in which we label specific regions of an image according to what is being shown. More specifically, the goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented. Because we're predicting for every pixel in the image, this task is commonly referred to as dense prediction or pixel-wise classification. Segmentation plays a central role in a broad range of applications, including medical image analysis, autonomous vehicles, video surveillance, change detection, city monitoring and augmented reality [27].

Since the goal is to classify each pixel of an image, semantic segmentation models takes an image as input and generates a corresponding mask as output, representing different labels with different colors for each pixel. Figure 2.1 shows an example of an input image for a semantic segmentation task with 5 five classes and its corresponding mask.

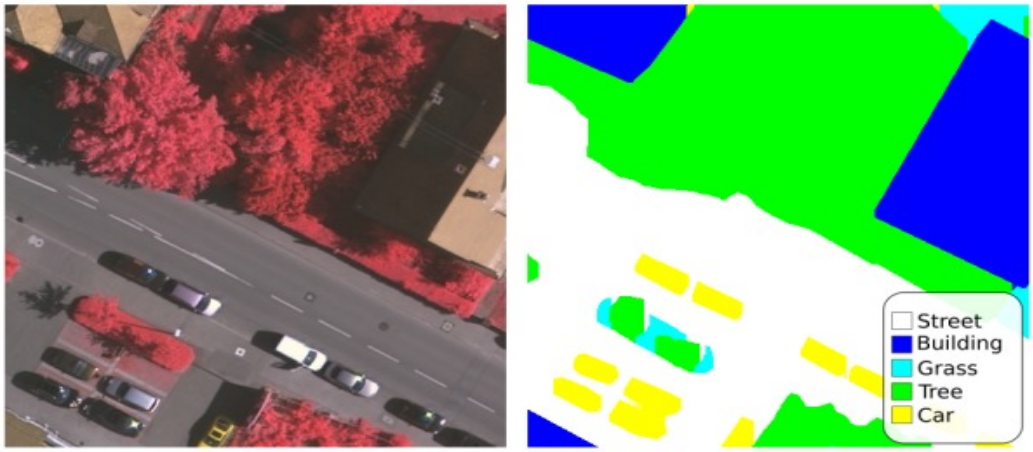

Figure 2.1: Example of an input image from the Potsdam dataset (on the left) and its corresponding segmentation mask (on the right).

As with other tasks, Deep Learning models have yielded a new generation of image segmentation models with great improvements of performance, often achieving the highest accuracy rates in popular benchmarks. [12] proposed one of the first CNNs based methods to perform semantic segmentation that achieved state-of-art results. Also tested for object detection tasks, their method - namely R-CNN - classify patches extracted from the images (using selective search algorithm). Their object detection system consists of three modules:

1. First module: generates a set of candidates detections to the detector, i.e category-independent region proposals.

2. Second module: extracts a fixed-length feature vector from each region.

3. Third module: a set of linear Support Vector Machine models, each on one for a class.

Later, [24] modified existing Convolutional Neural Networks architectures to manage non-fixed sized input and output, by replacing all fully-connected layers with the fully-convolutional layers. As a result, the model outputs a spatial segmentation map instead of classification scores.

[32] introduced the first U-Net architecture, which became widely known for its good performance. Basically, this type of network is composed of two parts, similar to an encoder-decoder scheme. The results of its applications for semantic segmentation of medical images and the speed and easiness of training a U-Net network as a end-to-end framework expanded the use of U-Net-based architectures for many fields of study. In its first application, U-Net surpass the previous state-of-art models in almost 10% in accuracy, achieving a 0.7756 Intersection over Union (IoU) score, almost 0.3 higher than the second method.

These architectures, among some other techniques - Generative Adversarial Networks (GAN's) - form the basis of the vast majority of models that represent the state-of-art for semantic segmentation. Various modifications have been made to adequate the pillar structure to the target problem, such as attention modules [37], relation modules [30] and skip-connections [46]. Although many of these results are very interesting, these networks are developed focused on a specific problem or field of study. Therefore, its mentions would escape the purpose of this work. In the next section we present papers much more related to our area of study.

2.2.1 Semantic Segmentation for Remote Sensing

Semantic Segmentation of Aerial images - often referred to as land cover classification - is essential in a wide range of fields, such as urban planning [38], crop and forest management [10], and disaster relief [11].

According to the authors themselves, [31] proposed the first work in the literature that perform pixel-wise semantic segmentation based on data-driven feature descriptors for high-resolution remote sensing images. The idea revolves around an adaptation of CNNs originally developed for classification tasks by a land-cover mapping method - called context windows - that assigns an object class to each pixel of an image by using sliding window. In a practical way, this methodology crops an input image in several parts with intersection areas. For each pixel, the method creates an image around it with a very smaller size than the original input and assign it the central pixel label to the entire image. The network is trained as a classification problem for the new images and at the end the corresponding mask is generated by taking the label of each image as a pixel class.

By inputting a new image for every pixel, context windows allows the network to interpret and extract features from the surrounding area, making the model capable of take into account the contextual information of each pixel. The authors tested the context windows methodology in two different remote sensing datasets: AGRICULTURE and URBAN, surpassing previous methods and concluding that their method is effective and robust for semantic segmentation of aerial images.

The context windows way of adapting a neural network to land-cover mapping tasks is a key point of this work and later, in the methodology section, we will delve deeper into it and elaborate how to adapt it to SMELL.

With a DenseU-Net based architecture, [9] proposes an end-to-end network with two inputs - SiameseDenseU-Net - that simultaneously uses both true orthophoto im-

ages and their corresponding normalized digital surface model as the input of the network structure. The experiments shows that a siamese network structure can be used to output a segmentation map for remote sensing data, overcoming some traditional state-of-art methods, as the original U-Net itself. The experiments also concludes that the SiameseDenseU-Net with a loss function weighted by the ratio of the median of the sample class frequency in the training set to the target class frequency, improves accuracy metrics by an average of 1% when compared with cross-entropy loss.

2.3 Deep Metric Learning for Semantic Segmentation of Aerial Images

Deep Metric Learning and Semantic Segmentation are hot trends in computer vision studies. DMeL methods to pixel-wise classification problems has been widely used, especially in the medical field. [19] successfully applied DMeL for Cell Nuclei segmentation using ResNet and DenseNet based architectures. The experiments reveal a 3.12% increase in the average value of Dice similarity coefficients using deep metric learning compared to no metric learning. [23] also accurate segmented biomedical images using DMeL. In their work, the author's method for lesion segmentation, reclassify the embedding vectors obtained from the deep metric learning model according to the distance and generate a enhanced segmentation result.

Although there are several works that use DMeL methods for semantic segmentation, its applications for remote sensing images still seem very little explored. [8] applied DMeL for hyperspectral imagery classification, which is also a pixel-wise classification task. The presented method was composed by a siamese structure with shared weights. The first network receives the original hyperspectral image as input, and for the other network, the authors used a low pass spatial filtering to introduce the spatial information. The outputs of each network from the siamese architecture are obtained with a hyperbolic tangent function and used to classify the land cover types respectively, and the final classification results are obtained by classification probability fusion.

[44] propose a novel DMeL approach to multi-hazard damage detection in remote-sensing images. Although the architecture was developed specifically for bitemporal data, the generated output is a land-cover mask, similar to semantic segmentation outputs, also classifying each pixel. In their work, a ResNet50-based triplet network discriminates the similarity between three inputs - anchor, positive and negative - by comparing the distance between them. The L2 distances between the embedded anchor and the other

two embedded inputs is applied to calculate a triplet loss function, optimized via back-propagation.

In brief, there are a lot of DMeL methods variations applied to the most diverse types of computer vision tasks. DMeL for Semantic Segmentation has already been proven an effective way to solve such problems. In remote sensing field, there still seems to be a lot to be explored. In this work, we tackle a very specific - and yet little explored - area of computer vision: Deep Metric Learning for Semantic Segmentation of Remote Sensing images.

Chapter 3

Background Concepts

This chapter presents important background concepts regards autoencoders for this work. We divided it in four sections: 3.1 refers to convolution, 3.2 explains pooling operations, 3.3 summarizes all the previous sections to explain the general flow of a convolutional autoencoder with subsections for its loss function and the optimization algorithm used in this work.

3.1 Convolution

The innovation of Convolutional Neural Networks (CNNs) is the ability to automatically learn a large number of filters in parallel, specific to a training dataset, under the constraints of a predictive modeling problem, such as image classification. The convolution operator allows filtering an input signal in order to extract some part of its content.

A convolution is a type of matrix operation, consisting of a kernel (or filter matrix), that slides over the input data performing element-wise multiplication with the part of the input it is on, then summing the results into an output [2]. The discrete convolution operation between an image i and a filter matrix w is defined as

$$h(x, y) = i(x, y) * w(x, y) = \sum_{n} \sum_{m} i(n, m) w(x - n, y - m), \tag{3.1}$$

where (x, y) are the coordinates to center the kernel w in the image t. Figure 3.1 illustrates how a convolutional operation works.

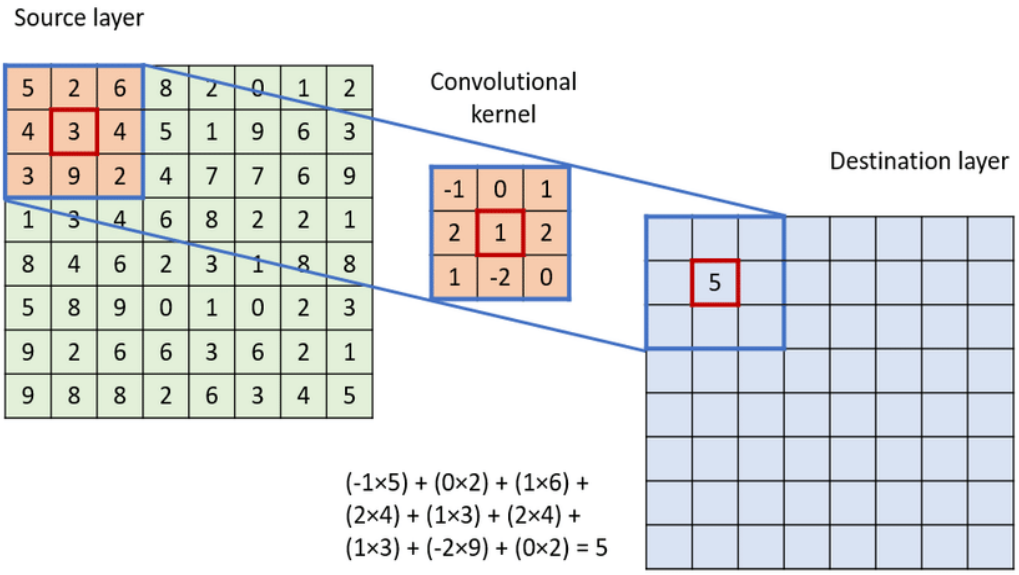

Figure 3.1: Convolution operation example. A dot product is calculated based on the 3x3 kernel w and the image f (source layer) and the result is appended into the destination layer.

There are two important parameters to be highlighted in convolution operations for CNNs: (1) *stride* concerns the spacing between kernel applications, i.e, the "jump size" for the kernel when sliding over the input; (2) *padding* which refers to replicating the image borders to create extra edges, making it possible for the kernel to have pixels near the edge of the image as the center pixel. The combination of these parameters may reduce the original dimension of the input data in the output.

3.2 Pooling

A major problem with convolutional layers is that the feature map produced by the filter is location-dependent. This means that during training, convolutional neural networks learn to associate the presence of a certain feature with a specific location in the input image. This can severely depress performance. Instead, we want the feature map and the network to be translation invariant - that means that the location of the feature should not matter.

In a convolutional neural network, pooling is usually applied on the feature map produced by a preceding convolutional layer and a non-linear activation function. The basic procedure of pooling is very similar to the convolution operation. A filter is select and slid over the output feature map of the preceding convolutional layer. Based on the

type of pooling operation selected, the pooling filter calculates an output on the receptive field. For average pooling, the mean value in each interaction between the kernel and the image is taken as output while max pooling takes the greater value and the min pooling takes the minimum value. Figure 3.2 shows an example of Max Poling with a (2×2) kernel size and (2×2) stride size, which means that there are no kernel overlaps in the sliding procedure.

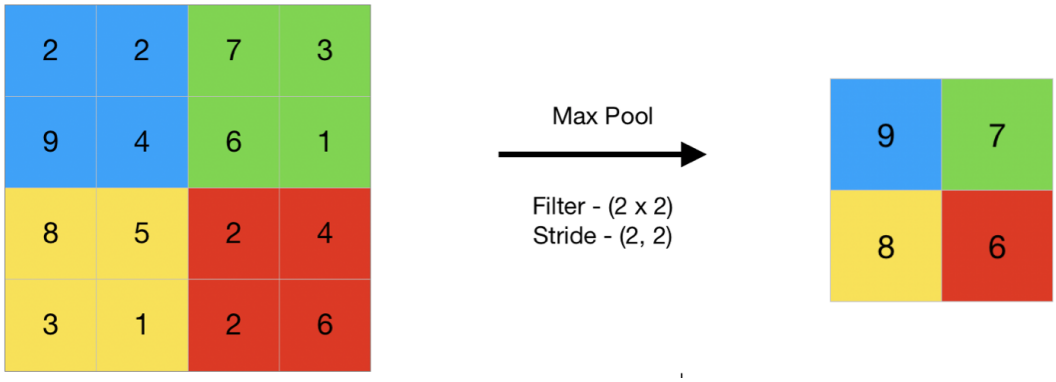

Figure 3.2: Pooling operation example. A kernel is slid over the output of the convolution and then a selected operation is applied. The figure shows an example of Max Pooling.

Note that pooling operations, like some convolutional operations, reduce the dimensions of the data. When a lot of layers of convolutional and pooling operations are stacked together (what is similar to an encoder network), the input dimension is drastically reduced into a new space. From this new space, it is possible to reconstruct the original input with high accuracy through a series of upsampling or transpose convolutional operations, what is defined as the decoder part of an autoencoder.

3.3 Convolutional Autoencoders

A Convolutional Autoencoder (CAE) is a type of Convolutional Neural Networks (CNNs) that is trained to reproduce its input image in the output layer. Since the input and output are the same images, this is not really supervised or unsupervised learning, so we typically call this self-supervised learning. An image is passed through an encoder, which is a CNN that produces a low-dimensional representation of the image. Rather than manually engineer convolutional filters we let the model learn the optimal filters that minimize the reconstruction error. Convolutional Autoencoders are the state-of-art tools for unsupervised learning of convolutional filters that can then be used in any computer vision task. Once these filters have been learned, they can be applied to any

input to extract features. These features can be used to do any task that requires a compact representation of the input, like classification.

Autoencoders in their traditional formulation [3] do not take into account the fact that a signal can be seen as a sum of other signals. Convolutional Autoencoders, instead, use the convolution operator to exploit this observation. They learn to encode the input in a set of simple signals and then try to reconstruct the input from them.

A Convolutional Autoencoder structure is defined by two parts. The first one, called encoder, is responsible for extract features from an input image through a series of convolutional and pooling operations. Once the input is encoded into a new space, often called *latent space*, the second part goes into action to reconstruct the original image from the latent space features, as shown in Figure 3.3.

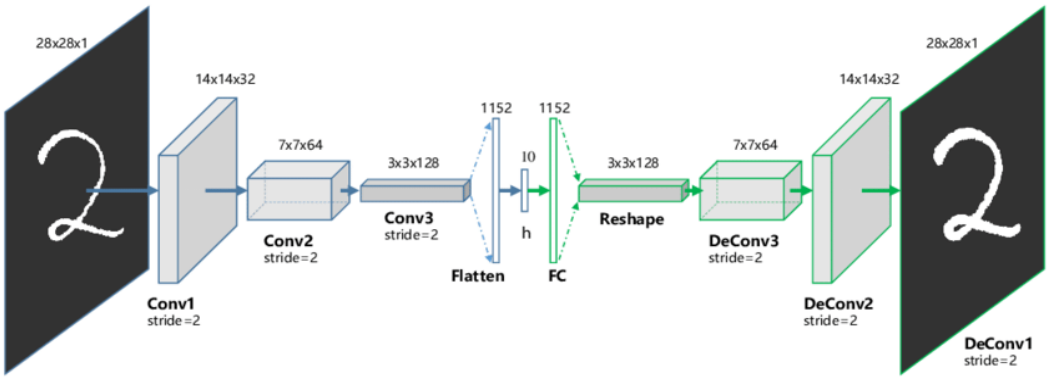

Figure 3.3: Convolutional Autoencoder architecture scheme for reconstructing a digit two image example.

In a very simple way, an autoencoder aims to learn a function $f_\Theta : X \to Z$, that encodes an input into the latent space and a function $f_{\Theta'}^{-1} : Z \to X$, that reconstructs the output from the features in the latent space. More directly, an autoencoder seeks to find a function $\psi_\Sigma : X \times X \to [0, 1]$, where $\Sigma = \{\Theta, \Theta'\}$. This last expression is the final mathematical formulation of the sentence: an autoencoder aims to reconstruct an image through its feature extraction with a encoder-decoder architecture.

An optimization algorithm, like Stochastic Gradient Descent (SGD), is used to find the optimal set of weights Σ^*, by minimizing a loss function, like Mean Squared Error (MSE). The following subsections briefly detail the role of the MSE loss function and the SGD algorithm.

3.3.1 Loss Function

Once the convolution and pooling layers are stacked together to define the encoder and the upsampling techniques - or the transpose convolutional operations - define the decoder, we have an autoencoder structure, as represented in Figure 3.3.

Since the main objective of an autoencoder network is to reconstruct an input after learning features from it, the element-wise loss function can be defined as $L(x_i, x_i') = ||x_i - x_i'||$ where x_i' is the output for the input x_i. For the general loss of autoencoders, the Mean Squared Error (MSE), is usually applied and is mathematically defined as $\Theta^* = \arg\min_{\Theta,\Theta'} \frac{1}{a} \sum_{x_i \in X} L(x_i, x_i')$, where a is the input length and Θ' is the autoencoder optimal weight set. The training of an autoencoder is performed through Backpropagation of the error, just like a regular feedforward neural network [17].

3.3.2 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a very popular and common algorithm used for optimizing loss functions in various Machine Learning algorithms and statistical estimations. Since the objective of the optimization module of a Neural Network is to minimize the loss function, SGD starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function.

The steps of the algorithm are:

1. Compute the gradient (∇) of the function.
2. Pick a random initial value for the parameters Σ.
3. Update the gradient function by plugging in the parameter values.
4. Calculate the step sizes s for each feature as : $s = \nabla \times \eta$, where η is the learning rate.
5. Calculate the new parameters as : $\Sigma' = \Sigma - s$.
6. Repeat steps 3 to 5 until gradient is almost 0 (smaller than a tolerance value ϵ).

More formally, Stochastic Gradient Descent update a parameter for each observation x_i and label y_i according to

$$\Sigma' = \Sigma - \eta \cdot \nabla_{\Sigma} L(\Sigma, x_i, y_i), \tag{3.2}$$

where η is the learning rate, L is the loss function and Σ denotes the parameters set of a machine learning model.

This approach performs frequent updates with a high variance that cause the loss function to fluctuate, which means that SGD can fluctuate around the minimum and might not converge to the minimum at all. Conversely, Stochastic Gradient Descent is able to jump to new and better local or global minima. One possible improvement for SGD is to decrease its learning rate over time. Another alternative is called Mini-batch Gradient Descent. It is a variation of SGD applied in this work and with a very similar the Gradient Descent method, as explict in Equation 3.3.

$$\Sigma = \Sigma - \eta \cdot \nabla_{\Sigma} L(\Sigma, x_{i:i+n}, y_{i:i+n}). \tag{3.3}$$

At each step, Mini-batch Gradient Descent computes the gradients on random sets of n observations, called mini-batches, instead of computing the gradients based on all observations or based on just one observation. This approach reduces the variance of the parameter updates and can get a performance boost in a distributed computing environment.

Chapter 4

SMELL for Semantic Segmentation of Aerial Images

This chapter details how the proposed method adapts SMELL for semantic segmentation tasks using the context windows technique to adequate the pairwise inputs and generate a land-cover mapping image as output. We indicate the modifications necessary for the adaptation to be successful as well as the parts of the original structure of SMELL that will remain intact. We go over SMELL's flow with the same notation originally used by [4]. The same is true for the context windows methodology and the mathematical notations used in [31]. We also explore the superpixel algorithm SLIC [1] for the final map refinement, in an attempt to group nearby and similar pixels and better define the segmented regions.

4.1 SMELL architecture

Supervised Distance Metric learning Encoder with Similarity Space (SMELL, [4]) simultaneously optimizes a latent data representation - using a DMeL model - and a similarity function that indicates the similarity of two objects in a latent space obtained through an siamese autoencoder and namely S-space. Originally tested to tackle image classification tasks, SMELL's architecture is composed of a siamese autoencoder with a shared set of weights, responsible for mapping the pairwise input into a latent space, and a metric learning module that aims to learn a metric from data and split similar from dissimilar images in the latent space. This both components of SMELL are trained together, as shown in Figure 4.1.

To optimize the similarity function metric learning algorithm of SMELL, markers are defined to represent one of two possibilities for the pairwise input, similar or dissimilar. The closer the inputs representation into the S-space from a marker that denotes similarity, the greater the probability that the inputs belong to the same class. The opposite is also

valid, the closer to a marker that represents dissimilarity, the greater the probability of inputs having different labels.

To better explain SMELL's framework, we split this section in three subsections. The first one explains the siamese autoencoder architecture, the second subsection explains how the metric learning module works and in the last part of this section we present detail about the loss functions, optimization and regularization.

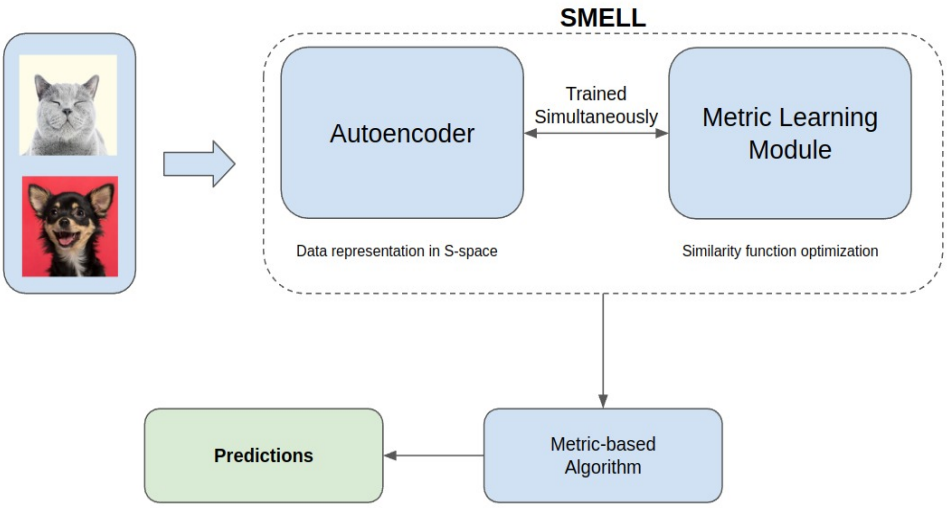

Figure 4.1: SMELL scheme. The siamese autoencoder receives two input images and optimizes the feature extraction and, consequently, the data representation into the latent space. The metric learning module optimizes a similarity function that separates similar from dissimilar objects. Lastly, the output of SMELL is used to feed a classifier that returns the predicted labels for the inputs.

4.1.1 Siamese Autoencoder

A siamese autoencoder can be viewed as two identical autoencoders with shared weights by the encoder parts. Mathematically, let the set $X = \{x_i\}_{i=1}^{a}$, with $x_i \in R^m$, be a input examples defined in an m-dimensional feature space with a correspondent label $y_i \in Y = \{y_i\}_{i=1}^{a}$ for each $x_i \in X$ with b possible labels, i.e $y_i \in \{1, ..., b\}$. An autoencoder structure aims to reproduce the input images by learning features into a latent space - encoder part - and then reconstructing the original input images from these features - decoder. A siamese autoencoder attempts to reconstruct two images at once. It means that it receives a pairwise input (x_i, x_j), extract a set of features from them sharing the same set of weights Θ, and returns a reconstructed output.

The first part of an autoencoder - the encoder - seeks to find a function $f_\Theta : X \to Z$ that maps the inputs into a new representation, in which $f(x_i) = z_i \Rightarrow l(x_i) = l'(z_i) = y_i$, where $l(x_i)$ and $l'(z_i)$ are functions applied to different representations of the input images that correspond to the input image label. The decoder does the opposite. It learns to map the new representation back to the original images, through a function $f^{-1}_{\Theta^1} : Z \to X$. More directly, a siamese autoencoder is a neural network that receives a pair of input examples $(x_i, x_j) \in X \times X$ and transform each of them to a latent data $(z_i, z_j) \in Z \times Z$ through a function f_Θ and then reconstruct the input from the latent data through a function $f^{-1}_{\Theta'}$. Figure 4.2 shows an example of a siamese autoencoder scheme.

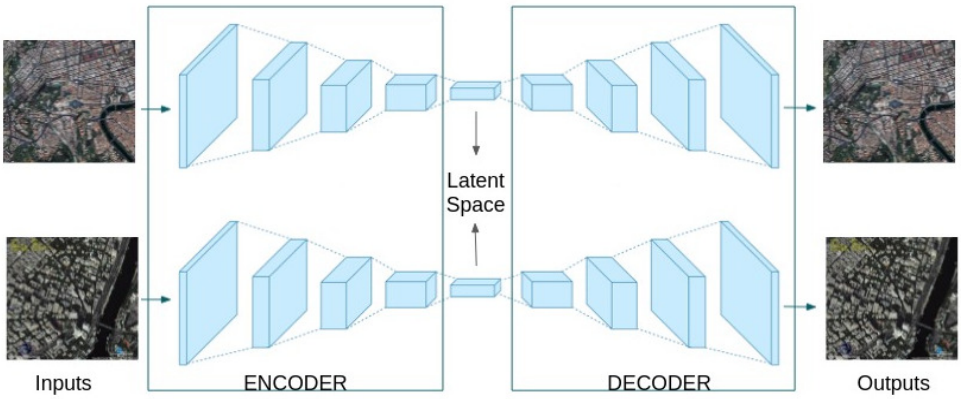

Figure 4.2: Example of siamese autoencoder neural network.

The new data representation space, called by [4] *latent feature space* - or just latent space - and defined as $Z = \{z_i\}^a_{i=1}$, with $Z \in R^n$, is the endpoint of the encoder and the input of the decoder. The inputs representation in this space is then used to build the proposed similarity space, or S-space through an L1-norm similar operation between the obtained inputs features.

4.1.2 Metric Learning and Similarity Space

Define (z_i, z_j) as the pairwise inputs (x_i, x_j) representation into the latent feature space $Z = \{z_i\}^a_{i=1}$. The *similarity space* is defined as $S = \{s_{ij}\}$ with $s_{ij} \in S \subset R^n$, where if $l(x_i) = l(x_j)$, then s_{ij} represents the similarity vector and if $l(x_i) \neq l(x_j)$ then s_{ij} represents the dissimilarity vector. The authors define a function f^S that maps the pairwise inputs (x_i, x_j) directly into the S-space as in Equation 4.1.

$$\begin{aligned} s_{ij} &= f^S(x_i, x_j) \\ &= |f_\Theta(x_i) - f_\Theta(x_j)| \\ &= |z_i - z_j| \\ &= (|z_i^1 - z_i^1|, |z_i^2 - z_i^2|, ..., |z_i^n - z_i^n|) \end{aligned} \tag{4.1}$$

In order to improve similarity calculations, a markers set $M \subset R^n$ is defined in the same space as S, as in Equation 4.2, consequently presenting the same dimension as the vector s_{ij}. Such markers are indicators of similarity/dissimilarity between the inputs, in space S, whose positions are randomly initialized and optimized together with the SMELL network using a cross-entropy loss function with a regularizer that avoid markers of the same representation - similarity or dissimilarity - from collapsing in the same area. Later, we detail such process with more details.

$$M = M^+ \cup M^- = \{\mu_i^+\}_{i=1}^k \cup \{\mu_j^-\}_{j=k+1}^w. \tag{4.2}$$

Two subsets of M, M^+ e M^-, represents the set of markers responsible for quantifying the similarity and the dissimilarity between the input pairs, respectively. Also, k is the number of positive markers and $w - k$ represents the number of negative markers (or markers that quantify dissimilarity) and $M^+ \cap M^- = \emptyset$. The closer the vector s_{ij} is to a marker $\mu^+ \in M^+$ or $\mu^- \in M^-$, the greater the probability that the elements of the pair (x_i, x_j) are similar or dissimilar to each other, respectively. To measure the similarity between s_{ij} and a specific marker $\mu_m \in M$, the authors use the Student's t-distribution with one degree of freedom as described in Equation 4.3.

$$q_{ij}^m = \frac{(1 + ||s_{ij} - \mu_m||_2^2)^{-1}}{\sum_{\mu_{m'} \in M}(1 + ||s_{ij} - \mu_{m'}||_2^2)^{-1}}, \tag{4.3}$$

where $q_{ij}^m \in R$ is the similarity/dissimilarity of s_{ij} in relation to the markers μ_m. Finally, to calculate the probability of $l(x_i) = l(x_j)$, all the positive markers similarity calculated are sum together. Parallel, the same goes for the probability of $l(x_i) \neq l(x_j)$ and the negative markers. Respectively, these probabilities are calculated as in 4.4.

$$\begin{aligned} q_{ij}^+ &= \sum_p q_{ij}^p, \quad \forall \mu_p \in M^+; \\ q_{ij}^- &= \sum_n q_{ij}^n, \quad \forall \mu_n \in M^-. \end{aligned} \tag{4.4}$$

Since M^+ and M^- are disjoint sets, we have $q_{ij}^+ + q_{ij}^- = 1$. To generate the correspondent label, these similarity/dissimilarity probabilities are then used as input to

a predictor algorithm, usually a classifier like K-Nearst Neighbors (K-NN), Support Vector Machine (SVM) or a simple Ordinary Least Squares (OLS) for linear binary cases. Note that similarity and dissimilarity contains two possible combinations for the input's labels each: $(0,0),(1,1)$ for similarity and $(0,1),(1,0)$ for dissimilarity. The final predictor aims to find the final class for each input image based on the difference between the inputs new representation, i.e, based on the inputs representation into the S-space, according to the markers similarity measures. More directly, it aims to identify which marker(s) represents which possible combination of labels.

4.1.3 Loss Function and Regularization

In this section, we explain how the loss function of SMELL works and the regularization methods applied to avoid *overfitting*. Since the autoencoder and the metric learning module are trained simultaneously in SMELL, we seek to find the set of parameters $\Sigma = \{\Theta, \Theta', M\}$ of the function $\psi_\Sigma(x_i, x_j)$, that defines the learned similarity function in space S. In the original paper, [4] estimate the optimal parameters $\Sigma*$ with cross-entropy loss H_c applied between the output of SMELL and object's classes and define regularization functions R_r and R_d applied to the autoencoder and the metric learning module, respectively, to avoid overfitting in the training process.

It is crucial that an autoencoder presents good generalization, i.e., that the produced representations yield low reconstruction error for both train and test samples. Regularized autoencoders limit the representational capacity of z provoking a bottleneck effect that does not allow the autoencoder to reconstruct the whole input and forces it to learn more meaningful features. As a consequence, it is trained to reconstruct well the training samples and also present small reconstruction error on test samples, implying generalization.

The first regularization function R_r - applied to the autoencoder - regards to reconstruction error. Hence, the authors proposal is based on regularization through the difference between the autoencoder inputs and outputs and the number of pairs in the training dataset with a constant r_r, a hyperparameter to be tuned as in 4.5.

$$R_r = \frac{r_r \sum_i \sum_j (||x_i - x_j||_2^2 + ||x_i' - x_j'||_2^2)}{N} \tag{4.5}$$

For the metric learning module, the authors note that when more than one marker that denotes similarity/dissimilarity is used, markers of the same set (M^+ or M^-) tend to group altogether, decreasing the efficiency of SMELL. To avoid such problem the authors

present a new regularization term R_d, called Repulsive Regularizer, for the proposed metric learning algorithm. Consistently to the markers methodology to calculate the probability of similarity/dissimilarity, this regularization function is composed of two opposite parts, one for positive markers R_d^+ and other for negative markers R_d^-, as defined in 4.6.

$$\begin{aligned} R_d^+ &= \frac{1}{c^+}\left[\sum_{\mu_i \in M^+}\sum_{\mu_j \in M^+}\frac{1}{||\mu_i - \mu_j||_2^2 + \epsilon}\right]; \\ R_d^- &= \frac{1}{c^-}\left[\sum_{\mu_i \in M^-}\sum_{\mu_j \in M^-}\frac{1}{||\mu_i - \mu_j||_2^2 + \epsilon}\right], \end{aligned} \tag{4.6}$$

where $\mu_i \neq \mu_j$, c^+ is a constant defined as the number of possible combinations of positive markers taken two by two and ϵ is a constant that prevents division by 0. After a grid search of experiments, the authors choose to use $\epsilon = 10^{-3}$. The final value of R_d is calculated as the sum of R_d^+ and R_d^- weighted by the parameter r_d, i.e $R_d = r_d(R_d^+ + R_d^-)$.

Let the SMELL output $Q = q_{ij}$ be the set that contains the pairs of similarity/dissimilarity probabilities $q_{ij} = (q_{ij}^+, q_{ij}^-)$. The optimal set of parameters is defined as $\Sigma^* = \arg\min_\Sigma J(X \times X)$. The cost function to be minimized is defined as in 4.7.

$$J(X \times X) = H_c(U||Q)r_{HC} + R_r + R_d, \tag{4.7}$$

where r_{HC} is a constant for calibration for the cross-entropy loss $H_c(U||Q)$. Seeking to find the optimal parameters set Σ^*, a mini-batch stochastic gradient descent (SGD) optimization method is applied among backpropagation. For training SMELL, randomly select the mini-batch with m pairs of elements, with half been similar, and the other half dissimilar.

4.2 Context Windows

In this section we dive deeper into the context windows methodology, proposed by [31]. We opted for the use of this methodology for two main reasons: (1) for being a simple method in its essence, it is easily applicable to any architecture developed to classification tasks - including SMELL - and (2) it shows the importance of contextual information to classify each pixel correctly. Adapting context windows to a DMeL method seems to be a very interesting perspective to explore semantic segmentation. We hope that

the contextual information around each pixel can be useful for SMELL's metric learning module to separate positive and negative markers more accurately.

As paired input for SMELL, we have two images with same dimensions $h \times w \times c$. It means that each image has a total of $h \cdot w$ pixels. Context windows method works as follows: for each pixel of each image we create a new image of size $N \times N \times c$ with the context around that pixel across every channel c, and label this new image with the same class as the central pixel. Figure 4.3 shows a visual scheme of how this works for a given image.

Figure 4.3: Illustration of context windows method for a pixel. It is replicated for all pixels in an image. The red square slides pixel by pixel over the entire image.

It is worth mentioning that the application of this technique results in an increase of data length, what is very limiting when working with high resolution images. For each image with dimensions $h \times w \times c$ we generate $h \cdot w$ images with dimensions $N \times N \times c$, what makes the algorithm for SMELL more computationally expensive for semantic segmentation tasks, creating a bottleneck according to the choice of the context window size N. Despite the increase in training time and memory allocation, context windows method still provides an easy and efficient way to adapt classification problems architectures into semantic segmentation tasks.

4.3 Simple Linear Iterative Clustering

Simple Linear Iterative Clustering, or SLIC [1], is a low computational cost algorithm that clusters pixels to efficiently generate compact, nearly uniform superpixels. The proximity of the pixels are also taken into account for clustering. Distant pixels can not belong to the same superpixel. As we can see in the example shown in Figure 4.4,

similar pixels are grouped together in a very efficient way, separating different color pixels in different superpixels.

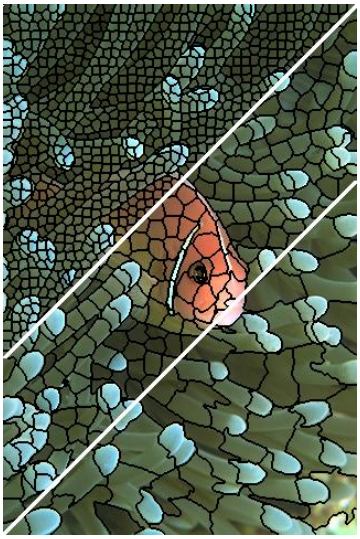

Figure 4.4: SLIC result image example with three different number of superpixels.

The algorithm segments an image according to a 5-dimensional distance metric comprised of spatial (x, y coordinates) and colour information (L, a, b, of the CIELAB colorspace) as shown in the following equations.

$$\begin{aligned} d_{lab} &= \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}; \\ d_{xy} &= \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}; \\ D_s &= d_{lab} + \frac{m}{S} d_{xy}, \end{aligned} \tag{4.8}$$

$\frac{m}{S}$ is a scaling factor where S is the initial cluster seed grid interval (dependent on image size and desired k) and m allows the user to control superpixel compactness and shape regularity. SLIC uses the same compactness parameter (chosen by user) for all superpixels in the image. If the image is smooth in certain regions but highly textured in others, SLIC produces smooth regular-sized superpixels in the smooth regions and highly irregular superpixels in the textured regions. So, it become tricky choosing the right parameter for each image [1].

The segmentation generated by SLIC is obtained by initializing K regularly spaced cluster centers and moving them to seed locations corresponding to the lowest gradient position in an $n \times n$ neighborhood to avoid noisy pixels as cluster center. Image gradients are computed as in 4.9.

$$G(x, y) = ||I(x + 1, y) - I(x - 1, y)||^2 + ||I(x, y + 1) + I(x, y - 1)||^2, \tag{4.9}$$

where $I(x, y)$ is the correspondent vector for the image in position (x, y). This takes into account both color and intensity information. Each pixel of the image is assigned to the nearest cluster center. After all the clusters are composed, the center and a residual error E are computed.

4.4 SMELL for Semantic Segmentation with Context Windows

The adaptation method to semantic segmentation applied to SMELL in this work is very straightforward. As previously mentioned, for an original aerial image with dimensions $w \times h \times c$, where w is the width, h is the height and c is the number of channels - three for RGB images - we have a total of $w \cdot h$ pixels. For each pixel p, we generate a new smaller image by cropping the original image with a predefined context window size N. Hence, the new images have dimensions $N \times N \times c$. These new images are then used as SMELL input, taken two by two, since SMELL is composed of a siamese autoencoder. After optimizing the feature extraction and the S space modules, SMELL's output is provided as input for a predictor, in our case an OLS predictor, that return the final class for each image. With the predicted labels, the original input is then reconstructed by taking only the central pixel predicted class - which is the same as the cropped image label - of the images generated by the context windows methodology. Figure 4.5 shows the workflow of our proposal.

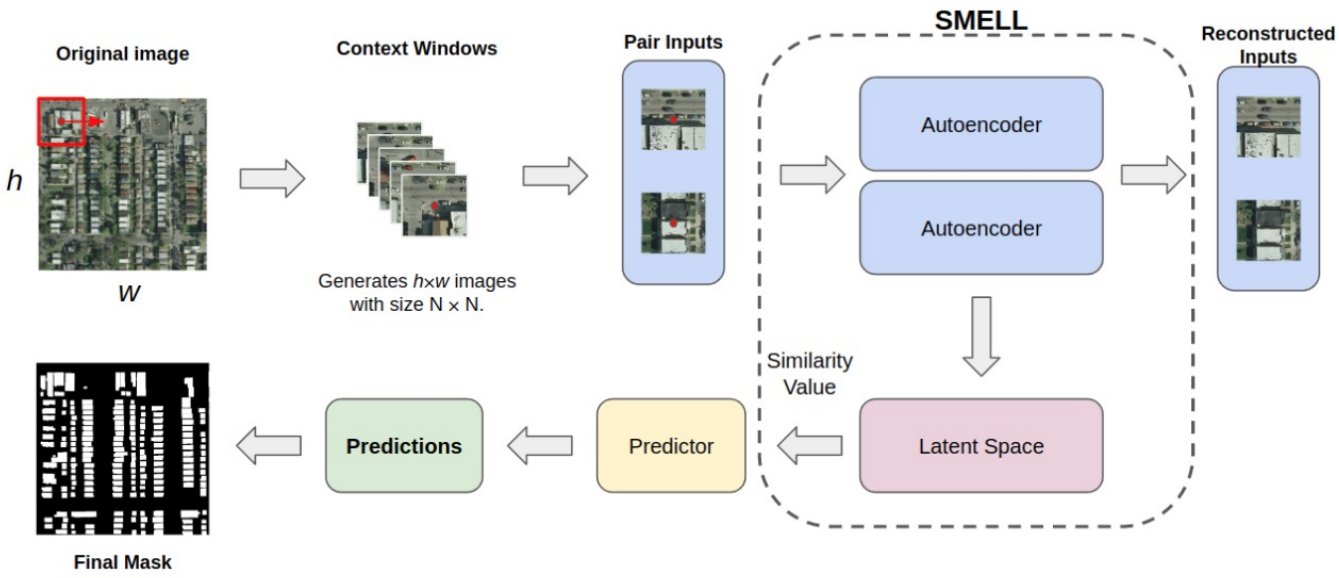

Figure 4.5: Scheme of SMELL's adaptation to semantic segmentation through context windows technique. The pair inputs are images obtained by sliding a context window of size N over the original image. These new images receives a label equals to its central pixel class (denoted as a red dot). SMELL is than trained as a classification network and the predictions are manipulated to compose the final mask - example for building segmentation.

Once the new cropped images are generated, a label equals to the class of the central pixel is assigned to the entire new image. Then, SMELL is trained as a siamese classification network. Inside SMELL, there are two main parts: the autoencoder and the S space. Both are trained simultaneously with the autoencoder's encoder aiming to learn features from the pairwise input from which the decoder can reconstruct the input images of SMELL - after the context windows has been applied - and the S space optimizing a representation space of the learned features using markers as reference.

Figure 4.6 visually demonstrates SMELL's processes with all the regularization functions. The siamese autoencoder with shared weights extracts relevant features for reconstruction through a series of convolution operations. With the encoded pairwise input, a $L1$ norm operation - Equation 4.1 - is applied to obtain s_{ij}, a representation of the features differences through all dimensions. This representation is optimize in the so called S space with the aid of similarity/dissimilarity markers, using cross entropy loss function with regularization.

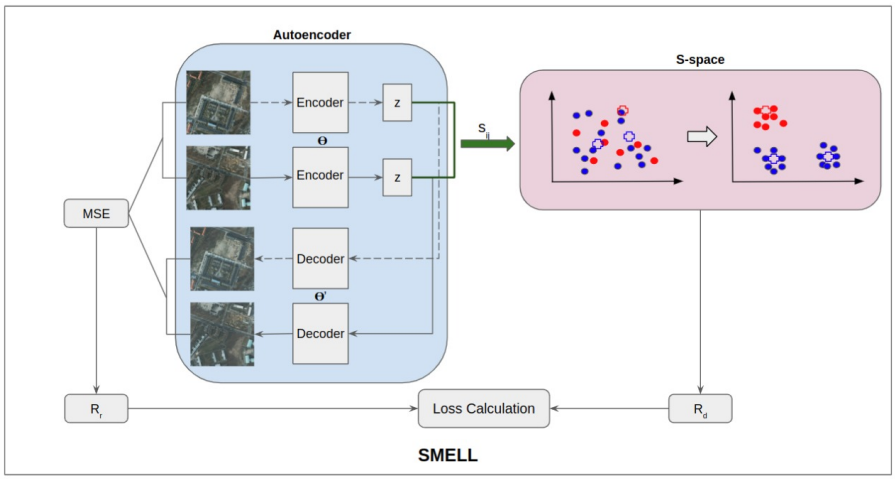

Figure 4.6: Scheme of SMELL's training. The pairwise input images go through a siamese autoencoder and the representation into the latent space is mapped to the S space where the probabilities of similarity and dissimilarity are calculated. The entire workflow is trained simultaneously with shared autoencoder weights.

Since we exploit a very little explored area, this works focus on evaluating SMELL's performance and measure the impact of variations in its components, such as the amount of contextual information provided to classify a single pixel (context windows size), the weights of the regularization functions and the autoencoder architecture. We also apply SLIC superpixel method on SMELL's output to check if it increases the accuracy or not. As baseline for comparisons, the context windows methodology was also applied in some state-of-art architectures, such as variations of ResNet and Xception. In the next chapter we present the experimental setup of this work as well as the datasets with which the experiments were performed.

Chapter 5

Experiments

The first section of this chapter brings a description of the experimental setup of this work, starting with the datasets employed in the experiments performed, along with the metrics used to analyze the results obtained and the parameters that were varied among the variation range. The second and third sections display these results and discuss them. All experiments were conducted using Python programming language (with tensorflow and Keras frameworks) and a NVIDIA GeForce GTX TITAN X GPU with 12GB of memory. In the evaluation sections, we present the mean results for a stratified 10-fold cross validation scheme with the standard deviation of each experiment in parenthesis.

5.1 Setup

This section covers the experimental setup used to test SMELL for semantic segmentation tasks using the context windows methodology as adaptation. We first introduce the datasets on which the networks were tested, followed by the chosen variations for the experiments. Contextual information is about the context window size, i.e the amount of information around each pixel that is taken into account as useful information for the central pixel classification, autoencoder depth subsection displays the variations made in the autoencoder architecture and regularization parameters shows the values tested when training SMELL.

5.1.1 Datasets

We tested SMELL's performance in three main datasets: Inria [26], Agriculture and Potsdam. All datasets are composed of aerial RGB images and contains a correspondent

mask for each image.

Inria

The Inria Aerial Image Labeling addresses the automatic pixelwise labeling of aerial imagery, a core topic in remote sensing. It contains ground truth data for two semantic classes: building and not building. More directly, the Inria dataset spans 5000×5000 images with a spatial resolution of 30cm per pixel on multiple cities across the globe and cover dissimilar urban settlements ranging from densely populated areas to alpine towns. The dataset was constructed by combining public domain imagery and public domain official building footprints. For this work, we selected five different cities as case for experiments: Austin, Chicago, Kitsap County, Tyrol and Vienna. Since each city has it own characteristics, such as population density (therefore different number of buildings per area), architectural styles and lighting, we consider each city as a different dataset with its own experiments.

Since the Inria dataset presents very high dimensional data - with 25 million pixels per image - and the context windows methodology generates one image for each pixel, we select only one half of the first image of each city as the training set - and the other half as validation set - to avoid a bottleneck due to memory capacity for the GPUs used. The first upper half of the second image of each city is used as test set. This bottleneck also impacts the training speed according to the original input images width and height. Later, even in this chapter, we discuss in more detail about this memory overhead caused by the inputs dimension. Figure 5.1 shows some examples of images from the Inria Dataset for Chicago, Kitsap and Vienna with the segmentation masks. It also illustrates the train and validation split over the Chicago example image.

Figure 5.1: Inria dataset examples with correspondent reference masks and train and validation split visual demonstration.

Agriculture

This dataset, also called Brazilian Coffee Scenes, is a composition of scenes taken by SPOT sensor in 2005 over four counties in the State of Minas Gerais, Brazil: Arceburgo, Guaranesia, Guaxupé and Monte Santo. It has 600×600 multi spectral high-resolution images of coffee crops and non-coffee areas. It has many intraclass variance caused by different crop management technique, as well as scenes with different plant ages and/or with spectral distortions caused by shadows. Unlike the Inria dataset case, the Agriculture dataset is composed of images of smaller dimensions, which allows it to use as many images as needed for achieving training convergence. However, training speed is still affected proportionally to the number of pixels. Figure 5.2 shows two examples of images from the Agriculture dataset, with their respective masks.

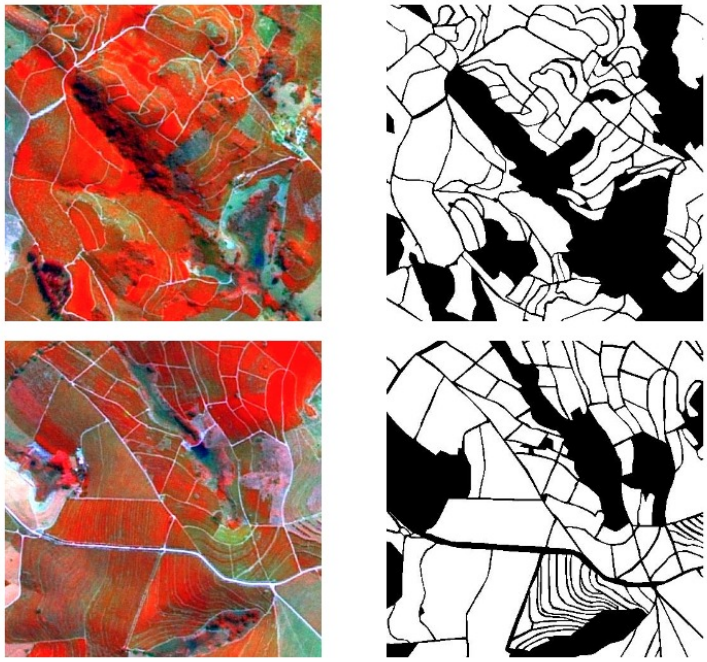

Figure 5.2: Agriculture Dataset examples with correspondent reference masks.

ISPRS Potsdam

The Potsdam dataset is a ISPRS labeled dataset for 2D semantic segmentation containing images patches of 6000×6000 pixels. The data were acquired using a ground sampling distance of 5 cm over Potsdam, Germany by BSF Swissphoto and segments the aerial images in six most common land cover classes: impervious surfaces, building, low vegetation, tree, car and clutter/background.

We explore the selected neural networks abilities of segmentation for two datasets with focus on trees and cars detection created from the original large-scale aerial images of the ISPRS Potsdam dataset. We aim to compare the performance of the selected architectures for the segmentation of different type of objects individually and also checking SMELL's ability to segment different items from aerial images. Figure 5.3 illustrates an example for the Potsdam dataset with a corresponding ground truth image. For this dataset, the bottleneck is also very present, similarly to the Inria dataset. We used the same approach of selecting only half of the first image for training, and the second half as validation.

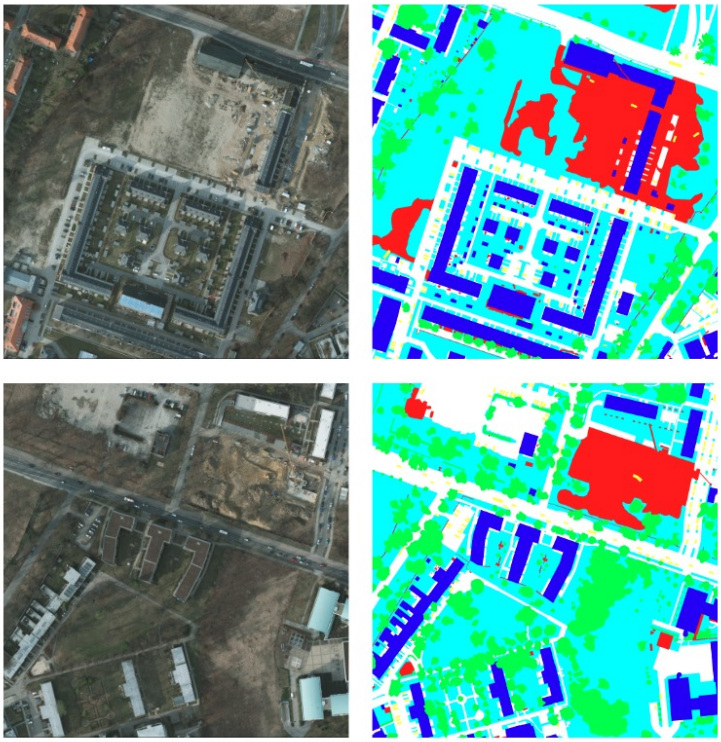

Figure 5.3: Potsdam Dataset examples with correspondent reference masks. Each color in the ground truth masks will be tested individualy, i.e a binary mask is generated for each color as black and all other pixels as white.

5.1.2 Contextual Information

In the most directly way possible, the amount of contextual information in our method means the size of the new generated image for each pixel of the original input. Therefore, the more contextual information the model uses, the bigger the new generated images from the context windows and, consequently, the worse the memory bottleneck will be. It creates a trade off for this methodology when using high resolution images, aiming to find the balance between the amount of images generated and the amount of contextual information. However, it only makes sense to use less input data in exchange for bigger cropped images if the increase of information around the objective pixel proves to be useful.

When the context windows methodology was proposed by [31], the authors tested it in two datasets: Agriculture and Urban. For the first one, the authors concluded that since coffee crops present homogeneous regions, larger context windows may bring more useful information. This proved to be true in the experiments, when using a 25×25

context window size overcame a 7×7 size. For the Urban dataset, the 7×7 context window input presented better results, due to the presence of more shuttered areas in urban regions.

In this work, we decided to explore a more continuous and better determined space of context windows sizes to measure if the increase of contextual information is more relevant for SMELL than for the chosen state-of-art baseline architectures. For each city of the Inria dataset we conduct a series of experiments with an amount of surrounding area that ranges from 15 to 21 with a step of 2.

For the Agriculture dataset, as the original input dimensions are significantly smaller than the Inria dataset imagery, it is possible to use more contextual information without extrapolate the maximum amount of memory. Thus, we tested a different range of context windows size: from 17 to 23 with a step of 2. As for the ISPRS Potsdam dataset, we only tested two context window sizes for each category: 7×7 and 21×21.

A table with the selected amount of contextual information to be tested for each dataset is presented below.

Table 5.1: Amount of contextual information used in tests for each dataset.

Dataset	**Context Windows Size**			
	7 x 7	**15 x 15**	**17 x 17**	**19 x 19**
Inria	No	[HTML]9AFF99Yes	[HTML]9AFF99Yes	[HTML]9AFF
Agriculture	No	No	[HTML]9AFF99Yes	[HTML]9AFF
ISPRS Potsdam	[HTML]9AFF99Yes	No	No	No

5.1.3 Autoencoder Depth

For the best results for context windows size obtained with SMELL, we vary the autoencoder depth in two different architectures. For the first one, called shallow autoencoder, the encoder part - responsible for extract features from the original inputs - consists of three groups of a convolutional layer and a max pooling operation. The number of convolutional filters in each group grows through the encoder. The first convolutional layer uses 32 filters, the second one 64 filters and the last convolution uses 128 filters. For the second encoder architecture tested, there are still three groups. However, each group is composed of two convolutional layers with the second layer having twice as many filters as the first layer. Figure 5.4 shows a representation of this two encoders variations. For all cases, we used a 4×4 kernel size for convolutions and a 2×2 kernel size for pooling operations. To maintain the original input dimensions, padding is applied in each convolution and pooling operations. The variation of the encoder's depth aims to explore

if more complex feature extractions through convolutions brings more useful information to better optimize SMELL's S space.

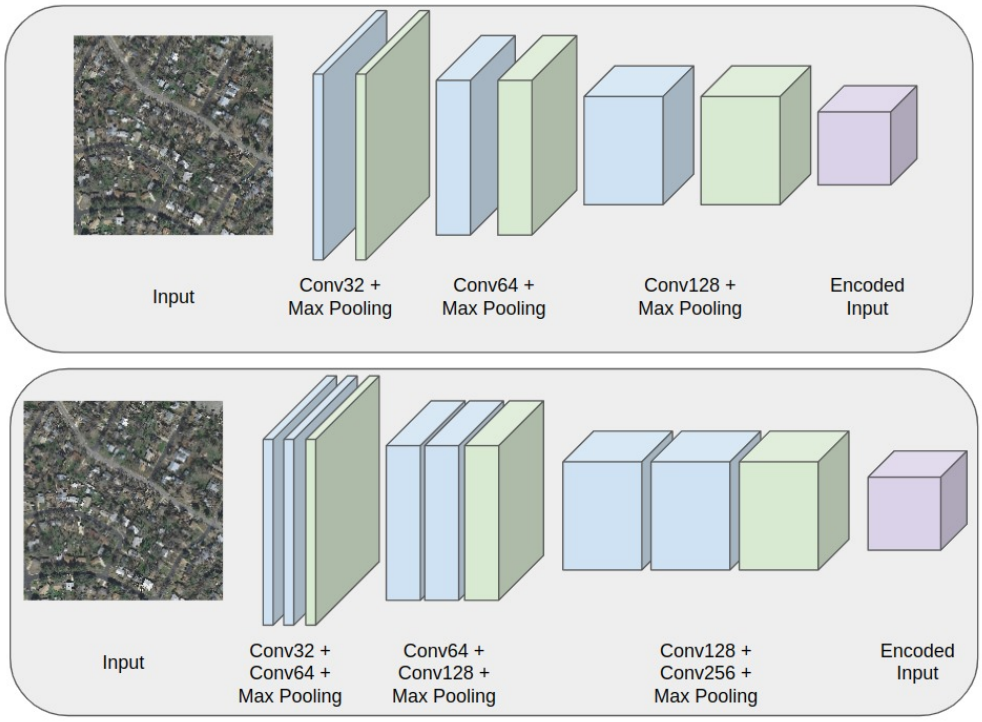

Figure 5.4: Two variations of encoder tested in SMELL's autoencoder. A shallow encoder with less convolutions layer (top) and a deep encoder with twice more convolutional layers (bottom).

Symmetrically to the encoder, two architectures of decoder were also tested. The shallow decoder is composed of three blocks of an upsampling layer followed by a convolution, where the first block has 256 convolution filters, the second has 126 filters and the last block convolution uses 64 filters. The deep decoder is composed of the same three blocks with the same pattern for the number of filters. However for each block, there are two convolutional layers for each upsampling. Combining the encoder and the decoder variations, we have a total of four variations for the autoencoder. We tested the autoencoder depth in the Inria dataset with Chicago images, aiming to find the best architectural setup for other experiments.

5.1.4 Metrics

For the validation of the evaluated methods, the following metrics will be applied:

1. Accuracy: describes how the model performs across all classes. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

2. F1-Score: a similar measure of the accuracy of a test. Takes into account both precision and recall in order to compute a score p, that is the number of correct positive results divided by the number of all positive results. It may be interpreted as a weighted average of the precision and recall, where an F-Measure reaches its best value at 1 and worst at 0.

3. Precision: the ratio between the number of positive samples correctly classified to the total number of samples classified as Positive. True positive cases over the sum of true positive and false positive cases. The precision reflects how reliable the model is in classifying samples as positive.

5.2 Evaluation in building segmentation

This section presents the results evaluation for semantic segmentation of building in aerial images using the Inria dataset. We compare SMELL against three state-of-art architectures: ResNet50, ResNet101 and Xception. We split this section in subsections exploring experiments with different purposes, comparisons and results, tracing an itinerary of building segmentation evaluations.

5.2.1 Autoencoder Depth

The first rounds of experiments aim to find the best autoencoder structure for our proposal. We selected four different architectures of autoencoder based on the number of convolutional layers in it. Selecting a network architecture base structure is a fundamental part of the training process, as it adapts strong pre-existing foundations in a more suitable way for each scenario. As metric for this experiment, we analyse only the F1-score. Since it combines the precision and recall of a classifier into a single metric by taking their harmonic mean it is primarily used to compare the performance of two architectures. Table 5.2.1 presents the results for the autoencoder depth analysis for Chicago city from Inria dataset, with a context windows size of 17.

Table 5.2: F1-score (with standard deviations) results for all four variations of autoencoder architecture.

	Encoder	
Decoder	**shallow**	**deep**
shallow	0.9018 (0.0215)	0.8975 (0.0234)
deep	0.8876 (0.0748)	0.8958 (0.0333)

The results show that a more complex architecture, with more features extracted by the encoder does not provide improvement in thew F1-score metric. This statement is also true for the decoder part. A more complex decoder, with more convolution operations to reconstruct the image, does not impact in the Mean Square Error (MSE) loss for autoencoder reconstruction task. A shallow encoder and a shallow decoder form the shallow autoencoder architecture applied to SMELL for all segmentation tasks. We selected the shallow autoencoder for running the further experiments for two main reasons:

1. The F1-score was higher for this setup: although the metric value for this structure was very similar to the others, a lower standard deviation value for the 10-Fold Cross Validation indicates a less volatile performance for a shallow autoencoder and more consistent results.

2. A shallow autoencoder is faster to train: less deep architectures, because they contain fewer convolutional layers and a smaller set of parameters to optimize, are computationally lighter and faster to train.

5.2.2 Inria dataset results

In this subsection we present the results regard all networks for all cities in the Inria dataset. We trained one network for each city in order to capture the architectural style and lightning variations. Table 5.1.2 summarizes the best results for each of the five tested cities of Inria dataset and all metrics using the test set.

First, the results show SMELL's ability to perform semantic building segmentation tasks in aerial images with high accuracy. Compared to other state-of-art architectures, SMELL's performance demonstrates a slight superiority in all metrics for Austin, Chicago and Vienna. For less populated cities, with less buildings to segment, the metrics demonstrate that the segmentation task performance is already saturated, with almost perfect accuracy for all networks. For this reason, we decided to use only the most heterogeneous cities in which SMELL surpass the baselines architectures for the further tests, such as

Table 5.3: Best results in all metrics for each city in the Inria dataset with standard deviation.

City	Method	Accuracy	F1-Score	Precision
Austin	**ResNet50**	0.9119 (0.0098)	0.9047 (0.0124)	0.9055 (0.0124)
	ResNet101	0.913 (0.0114)	0.9077 (0.0135)	0.9066 (0.0139)
	Xception	0.9210 (0.0103)	0.9175 (0.0109)	0.9167 (0.0114)
	SMELL	**0.9276 (0.0151)**	**0.9294 (0.0136)**	**0.9329 (0.0117)**
Chicago	**ResNet50**	0.8861 (0.0207)	0.8826 (0.0235)	0.8832 (0.0240)
	ResNet101	0.8864 (0.0142)	0.8836 (0.0157)	0.8832 (0.0167)
	Xception	0.892 (0.0240)	0.8898 (0.0253)	0.8904 (0.0259)
	SMELL	**0.9003 (0.0291)**	**0.9022 (0.0278)**	**0.9068 (0.0259)**
Kitsap	**ResNet50**	0.9978 (0.0011)	0.9976 (0.0010)	0.9977 (0.001)
	ResNet101	0.9976 (0.0010)	0.9975 (0.0008)	0.9978 (0.0005)
	Xception	0.9980 (0.0007)	0.9979 (0.0007)	0.9980 (0.0007)
	SMELL	**0.9980 (0.0015)**	**0.9982 (0.0011)**	**0.9985 (0.0006)**
Tyrol	**ResNet50**	0.9787 (0.0071)	0.9787 (0.0057)	0.9799 (0.004)
	ResNet101	**0.9796 (0.0048)**	**0.9791 (0.0042)**	**0.9797 (0.0037)**
	Xception	0.9766 (0.0100)	0.9768 (0.0081)	0.9784 (0.0059)
	SMELL	0.9768 (0.0128)	0.9786 (0.0100)	0.9822 (0.0049)
Vienna	**ResNet50**	0.9139 (0.0115)	0.9120 (0.0114)	0.9124 (0.0118)
	ResNet101	0.6390 (0.0172)	0.5944 (0.0154)	0.5838 (0.0216)
	Xception	0.9188 (0.0105)	0.9177 (0.0106)	0.9177 (0.0108)
	SMELL	**0.9253 (0.0150)**	**0.9273 (0.0139)**	**0.9318 (0.0112)**

SLIC superpixel application and autoencoder depth. The following figures illustrate the best results of each network for each city.

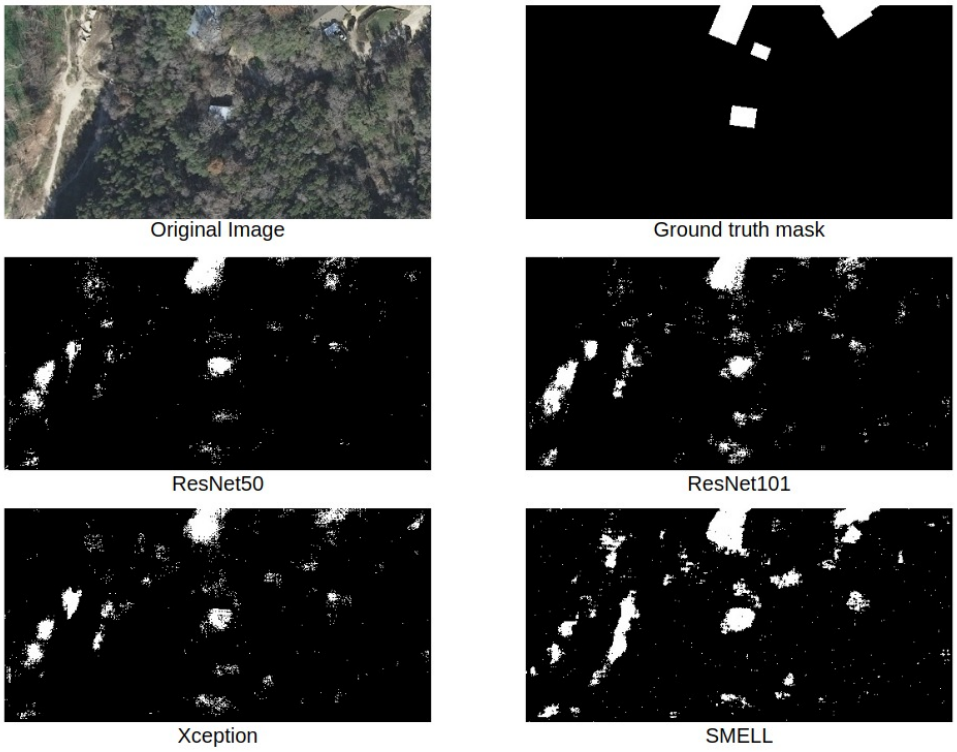

Figure 5.5: Land cover masks results for each network with correspondent original input and ground truth mask for Austin city.

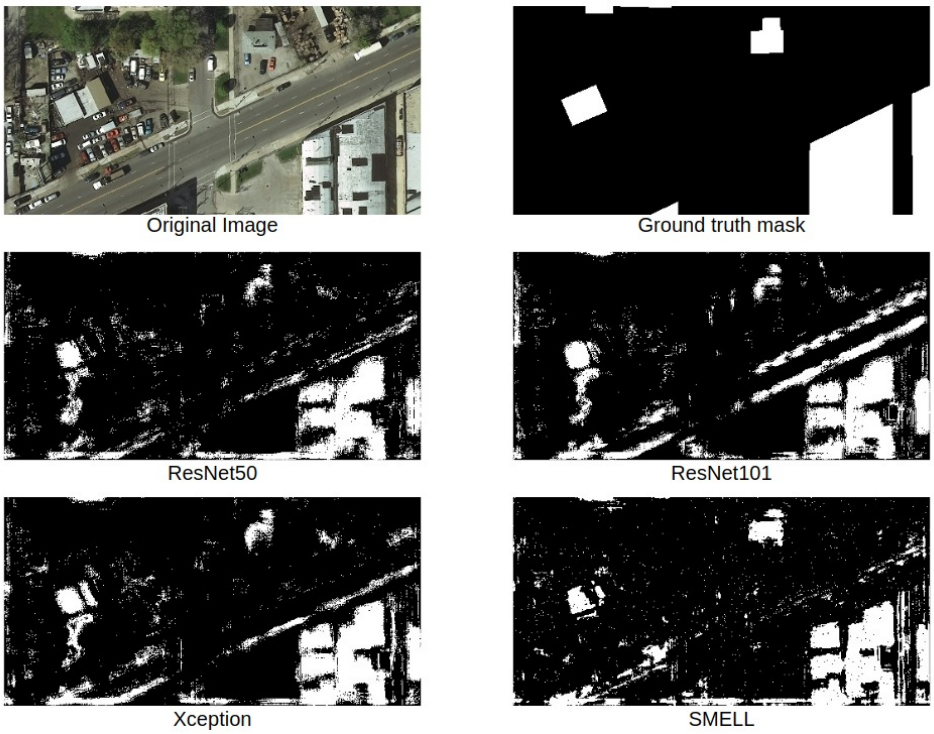

Figure 5.6: Land cover masks results for each network with correspondent original input and ground truth mask for Chicago city.

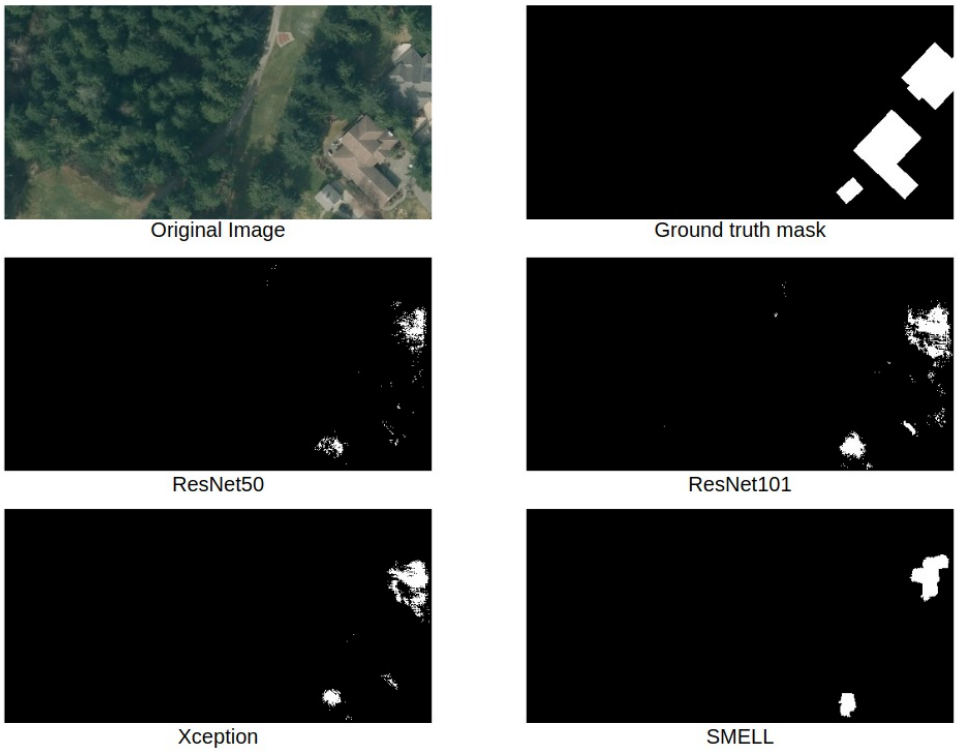

Figure 5.7: Land cover masks results for each network with correspondent original input and ground truth mask for Kitsap County.

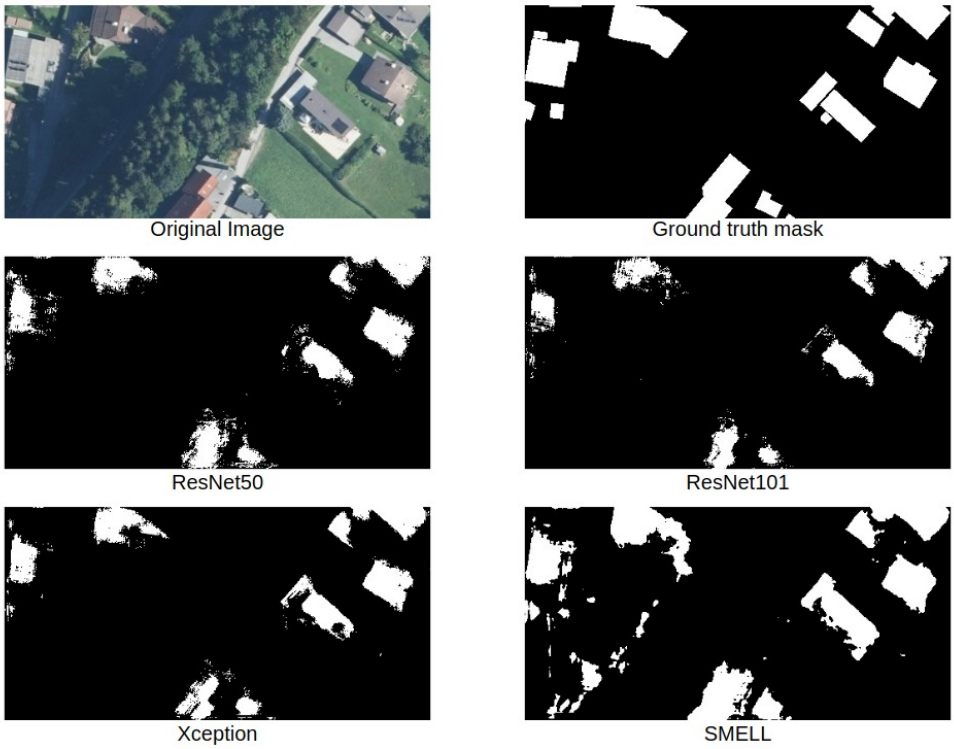

Figure 5.8: Land cover masks results for each network with correspondent original input and ground truth mask for Tyrol city.

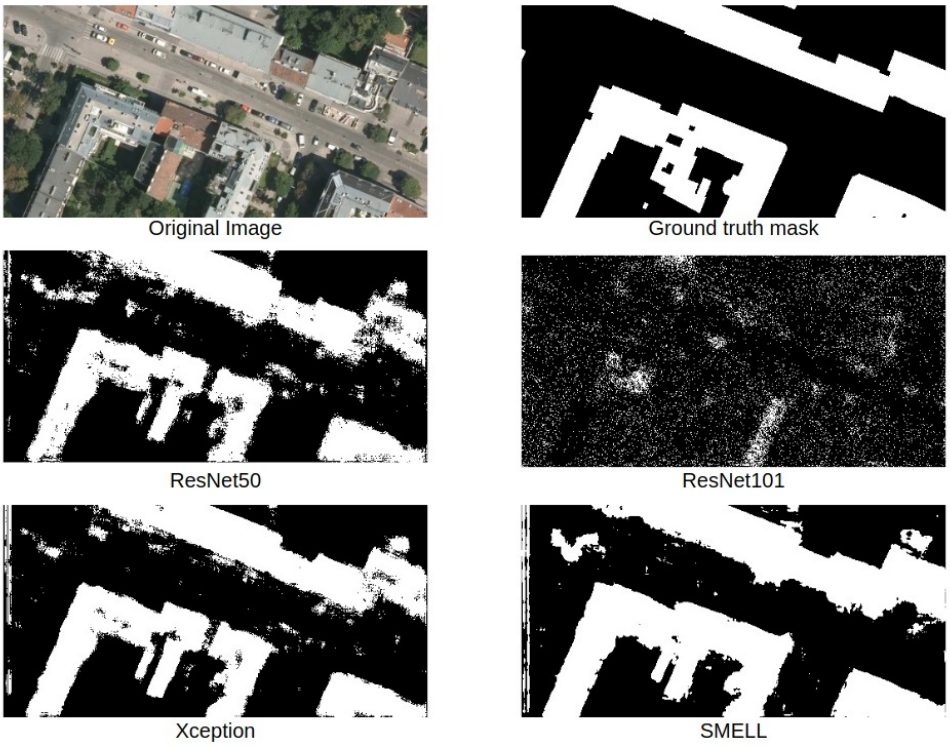

Figure 5.9: Land cover masks results for each network with correspondent original input and ground truth mask for Vienna city.

The visual results corroborate with the metrics values: SMELL is capable of detecting building pixels at least as well as state-of-art architectures. As for misclassified pixels, those with colors similar to the buildings pixels seem to make up the majority of misclassifications for all networks. The cases where no building labeled pixels presents coloring similar to pixels vary from city to city and from image to image. For all tested networks - SMELL included - the generated masks show the difficulty of separating sidewalks and buildings pixels for the Chicago example, that clearly presents misclassified pavement pixels as building with SMELL being the network that least confuses these cases. For Tyrol example image, SMELL misclassified pixels in tree shadows as building pixels more often the the other networks. However for true positive cases (building pixels), our proposoal surpasses the other networks, segmenting building areas more accurately.

Although SMELL performs well for building segmentation, some cases can still be improved, such as in Austin example. We notice that the misclassified pixels occur in a spread way and not in a concentrated area. In order to add more importance to the neighbourhood pixel classes, we use SLIC superpixel aiming to reduce this isolated misclassified pixels cases and increasing SMELL's accuracy.

5.2.3 Context Windows Size

As detailed in the previous section, we vary the context windows size and the autoencoder depth to find the best set up for SMELL and the baselines networks. For all five cities mentioned for the Inria dataset, we tested the four different context windows size, ranging from 15×15 to 21×21 with a step of 2, to check the networks response to many levels of contextual information. Table 5.2.3 presents the accuracy results for all tested networks and context windows size in all cities selected from Inria dataset.

Note that contextual information displays a significantly more important role for SMELL than for current state-of-art models. Although almost all networks have demonstrated a performance increase for larger context windows sizes, SMELL's accuracy is slightly lower than for Xception with smaller context windows size and, when more neighbourhood area is provided to classify each pixel, SMELL outperforms all baseline networks for cities with more densely populated areas, such as Austin, Chicago and Vienna. Next, we display visual examples for some cities and networks, among different context windows size.

Table 5.4: Accuracy results for the Inria dataset with different context sizes.

City	Architecture	Context Windows Size			
		15 x 15	17 x 17	19 x 19	21 x 21
Austin	**ResNet50**	0.9049	0.9095	0.9108	0.9119
	ResNet101	0.9083	0.9175	0.9130	0.9130
	Xception	0.9142	0.9152	0.9192	0.9210
	SMELL	0.9010	0.8987	**0.9098**	**0.9276**
Chicago	**ResNet50**	0.8788	0.8851	0.8859	0.8861
	ResNet101	0.8787	0.8849	0.8864	0.8845
	Xception	0.8913	0.8898	0.8913	0.8920
	SMELL	0.8920	0.8858	**0.9003**	0.8993
Kitsap	**ResNet50**	0.9965	0.9975	0.9977	0.9978
	ResNet101	0.9944	0.9976	0.9975	0.9972
	Xception	0.9978	0.9980	0.9980	0.9978
	SMELL	0.9981	0.9979	**0.9981**	0.9978
Tyrol	**ResNet50**	0.9761	0.9785	0.9787	0.9779
	ResNet101	0.4997	0.4754	0.9795	**0.9796**
	Xception	0.9775	0.9760	0.9766	0.9764
	SMELL	0.9575	0.9686	0.9711	0.9768
Vienna	**ResNet50**	0.9035	0.9104	0.9117	0.9139
	ResNet101	0.6305	0.6473	0.6390	0.5750
	Xception	0.9126	0.9150	0.9174	0.9188
	SMELL	0.9102	0.9170	0.9193	**0.9253**

Figure 5.10 shows an example of predicted masks for each context windows size

tested. Smaller sizes of context generates worst segmentation results. Misclassified pixels, such as pavement cases, are significantly reduced when bigger context windows is applied. It is also noticeable that building shapes become more precise with the use of more contextual information, generating very similar results to the ground truth mask.

However, there are still misclassified pixels spread around the image. To correct this cases and improve even more the achieved results, we apply SLIC superpixel, to group nearby pixels according to the majority labels. We vary the principal parameters of SLIC seeking to find the best configuration for each case. The next subsection presents the results regards SLIC applications.

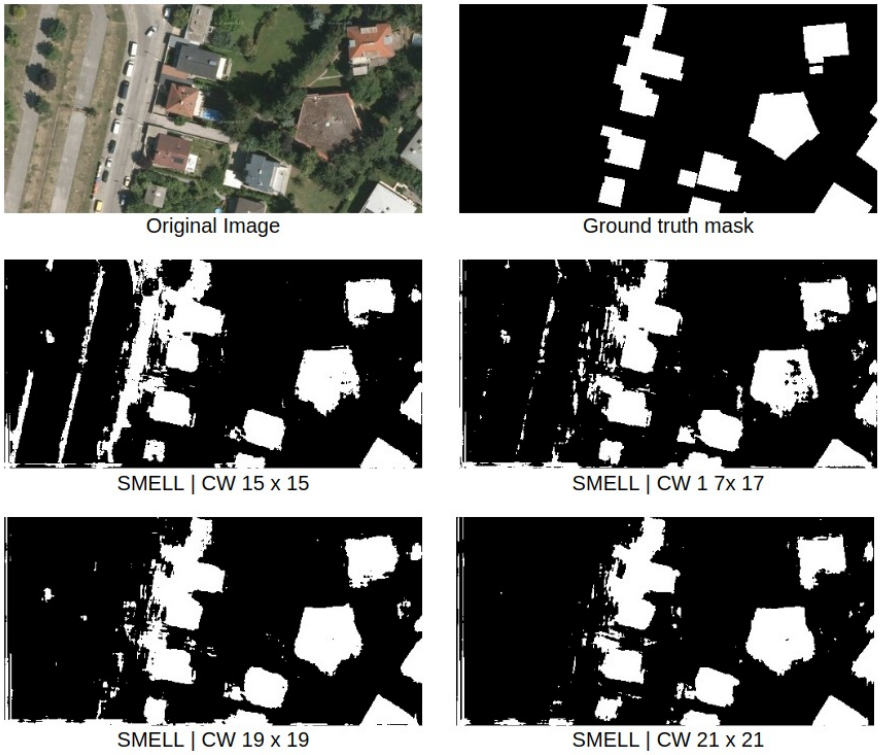

Figure 5.10: Vienna example for SMELL with all context windows. Higher context windows size improve the power of segmentation for SMELL.

5.2.4 Superpixel

Based on the results shown, the presence of misclassification for some isolated pixels is notable. To achieve better results we applied SLIC superpixel methodology to try to collapse nearby pixels as the majority label. This process also takes into account the contextual information and can be seen as a correction step for SMELL's wrong predictions.

As presented in the SLIC methodology section, there are two main parameters in SLIC: the number of superpixels - or number of segments - and the compactness. We tested the performance of a wide range of combinations for these parameters. Figure 5.11 presents a visual interpretation of SMELL's performance for each combination for Austin, Chicago and Vienna cities.

Firstly, SLIC seems to improve SMELL's output when compared to the original ground truth mask for all city cases. For Chicago, the 200 segments and 120 compactness SLIC achieved the best performance, with 0.9299 accuracy. For Vienna, the best setup was SLIC with 100 segments and compactness equals to 40, achieving 0.9222 accuracy and. Lastly, for Austin city, SMELL with SLIC achieved 0.9879 accuracy, a great improvement in the original 0.9276 accuracy value.

The graph shows that the best results are obtained with a small number of segments and high compactness for Austin, while SLIC performs at its worst for Chicago with 100 segments. Using the best setup found for each city, we generate some visual examples of SMELL's building segmentation performance to compare with the ground truth mask and with SMELL's performance without SLIC. Figure 5.12 shows an example for each of the tested city. As noticed, there is no right configuration for SLIC. However, the results suggest that a small number of segments is more suitable for images with more buildings spread out in more uniform spaces, since there is no need for smaller segments once pixels from one building unit are very close to each other and very apart from other building units pixels. This behaviour is found in the Austin example, where the number of segments with best performance for SLIC is 100. Inversely, for images with more buildings pixels, such as the Vienna example, for higher values of compactness, a higher number of segments seems more appropriate. Curiously, there is one combination - 100 segments with compactness equals to 40 - that surpasses all others, similar to an outlier.

The generated masks outputs shows that, despite the good metrics, SMELL presents a greater imprecision to classify the building borders area, making the shape of the buildings of the generated masks not as assertive as in the ground truth masks. This creates a large set of misclassification pixels spread out almost evenly over the image that is fixed by SLIC application in SMELL's output mask. In general, SLIC visual results demonstrate agreement with the increase in metrics, adjusting each pixel label based on the nearby pixels classes and better defining building and no building areas. The results also shown that SMELL performs better for classifying building pixels than for no building pixels, what leads to better accuracy for more densely populated areas.

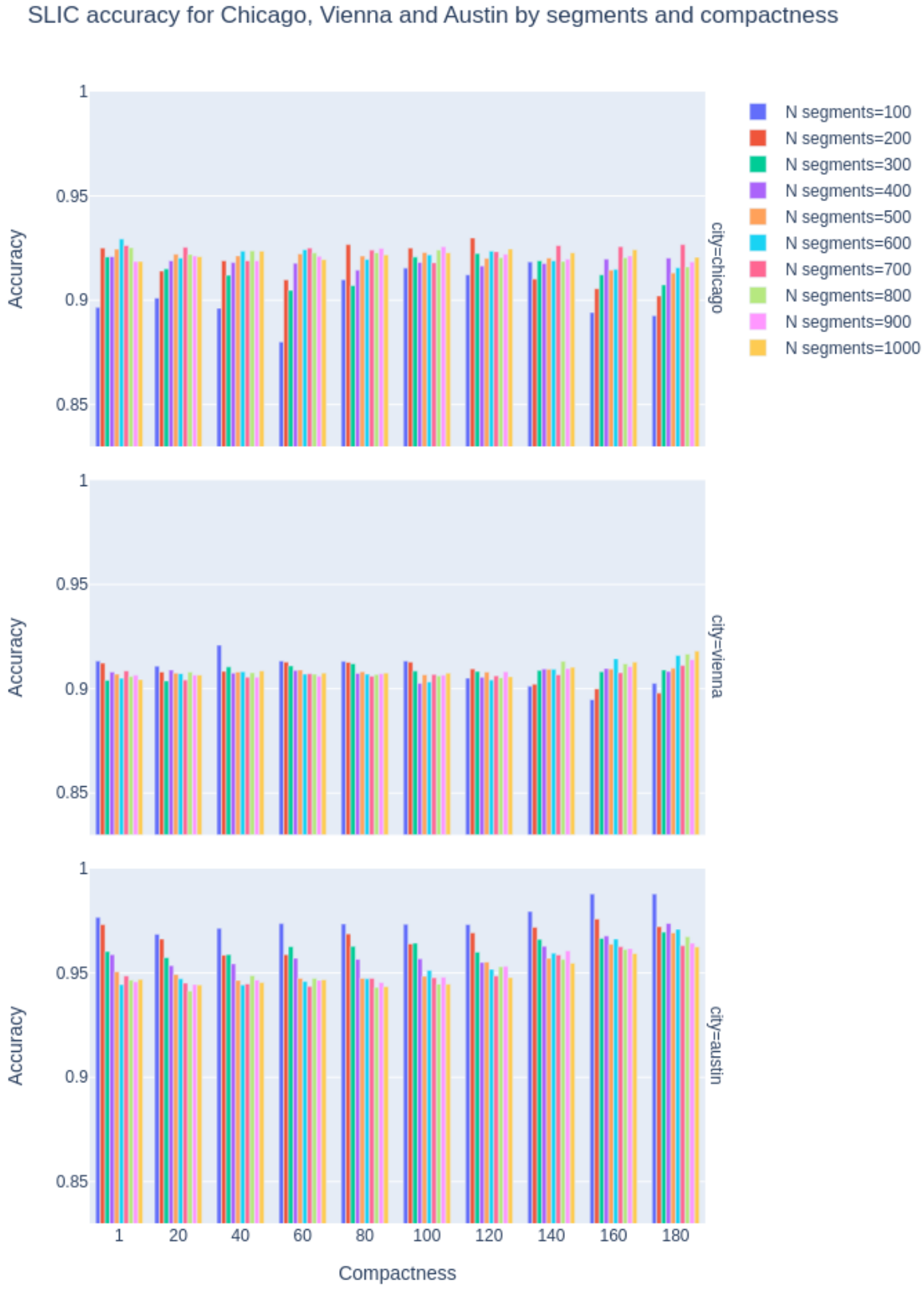

Figure 5.11: SLIC accuracy results for different number of segments - from 1 to 180 with a step of 20 - and compactness - from 100 to 1000 with a step of 1000. Best results are found using a small number of segments with a high compactness combination.

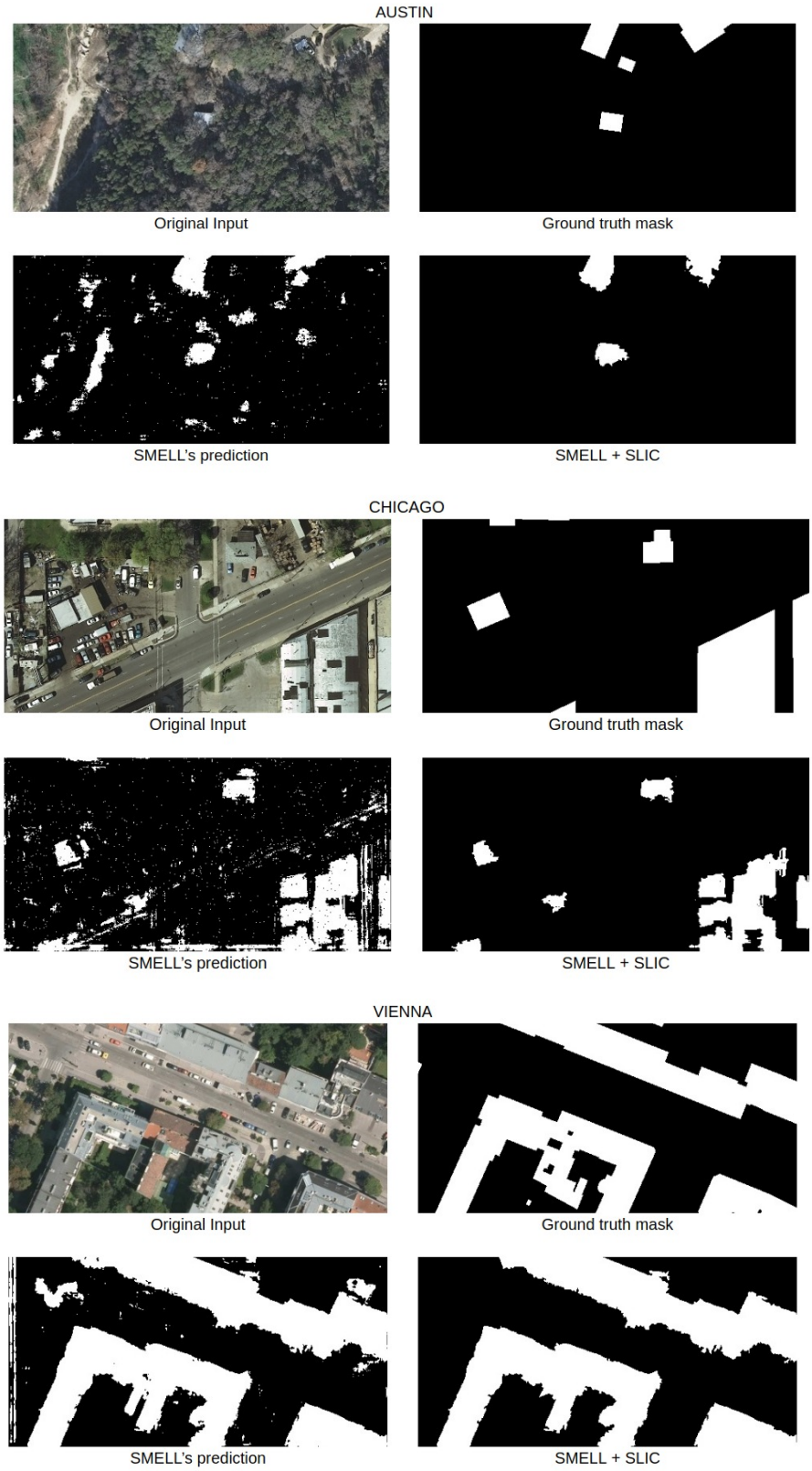

Figure 5.12: An example of an input image with (left top) its corresponding mask (right top) and the segmentation output from SMELL (left bottom) and from SMELL with SLIC (right bottom) for Austin, Chicago and Vienna cities.

5.3 Evaluation in coffee crops segmentation

This section presents the results evaluation for semantic segmentation of Brazilian coffee scenes dataset. We follow the experimental setup defined earlier comparing SMELL performance against ResNet50, ResNet101 and Xception. Unlike the methodology used for the Inria dataset, we performed a more direct set of experiments, since this dataset contains smaller images and no partition as the cities in Inria.

In addition, previous experiments using context windows for this dataset conducted by [31] have already established a path to be followed. The conclusion that larger context windows perform better for segmenting coffee plantations led us to test the range from 17 to 21 without extrapolating the memory capacity due to the bottleneck caused by the generation of new images. For the 23×23 context window case, the amount of memory needed exceed the available. To get around this problem, we applied a stride of 2 in the context windows slide operation - flagged as $S = 2$. In practice, it means that we use only half of the images for the highest context windows size. Despite it generates some nonconformity for the experiments, it is interesting that even with less images available to train, the performance metrics were greater for large contextual information, meaning that the contextual information can be considerable more important for pixel wise classification when enough data is provided to the model.

The following tables show the average results for the various contextual information amount tested over all metrics with standard deviation for a 10-Fold Cross Validation procedure.

Table 5.5: Brazilian Coffee Scenes dataset results with a 17×17 context windows size.

Method	Accuracy	F1-Score	Precision
ResNet50	0.7236 (0.0162)	0.7279 (0.0151)	0.7359 (0.0138)
ResNet101	0.7118 (0.0053)	0.7168 (0.0050)	0.7253 (0.0044)
Xception	0.8066 (0.0037)	0.8055 (0.0033)	0.8048 (0.0031)
SMELL	**0.8452 (0.0096)**	**0.8411 (0.0086)**	**0.8432 (0.0102)**

Table 5.6: Brazilian Coffee Scenes dataset results with a 19×19 context windows size.

Method	Accuracy	F1-Score	Precision
ResNet50	0.8066 (0.0072)	0.8039 (0.0060)	0.8033 (0.0061)
ResNet101	0.8199 (0.0052)	0.8167 (0.0040)	0.8166 (0.0045)
Xception	0.8121 (0.0046)	0.8105 (0.0041)	0.8097 (0.0040)
SMELL	**0.8511 (0.0104)**	**0.8463 (0.0094)**	**0.8507 (0.0130)**

Table 5.7: Brazilian Coffee Scenes dataset results with a 21×21 context windows size.

Method	Accuracy	F1-Score	Precision
ResNet50	0.8083 (0.0024)	0.8053 (0.0019)	0.8044 (0.0020)
ResNet101	0.8203 (0.0054)	0.8163 (0.0050)	0.8164 (0.0056)
Xception	0.8229 (0.0050)	0.8205 (0.0041)	0.8199 (0.0044)
SMELL	**0.8511 (0.0098)**	**0.8469 (0.0092)**	**0.8495 (0.0112)**

Table 5.8: Brazilian Coffee Scenes dataset results with a 23×23 context windows size.

Method	Accuracy	F1-Score	Precision
ResNet50	0.8152 (0.0052)	0.8114 (0.0047)	0.8111(0.0051)
ResNet101	0.8218 (0.0045)	0.8179 (0.0035)	0.8182 (0.0045)
Xception	0.8291 (0.0118)	0.8266 (0.0099)	0.8272 (0.0106)
SMELL	**0.8588 (0.0097)**	**0.8539 (0.0093)**	**0.8588 (0.0117)**

As can be seen from the results obtained, SMELL surpasses the state-of-art networks in all scenarios, proving that distance metrics can be a useful tool for semantic segmentation tasks. Furthermore, even in a better defined space for the context windows size, larger context windows produce better results for all tested networks. In the test stage of these experiments, ResNet50 and ResNet101 showed greater difficulty in training adjustments, with tendencies to overfitting more easily than SMELL or Xception. Next, Figure 5.13 presents visual results examples for each network best accuracy performance.

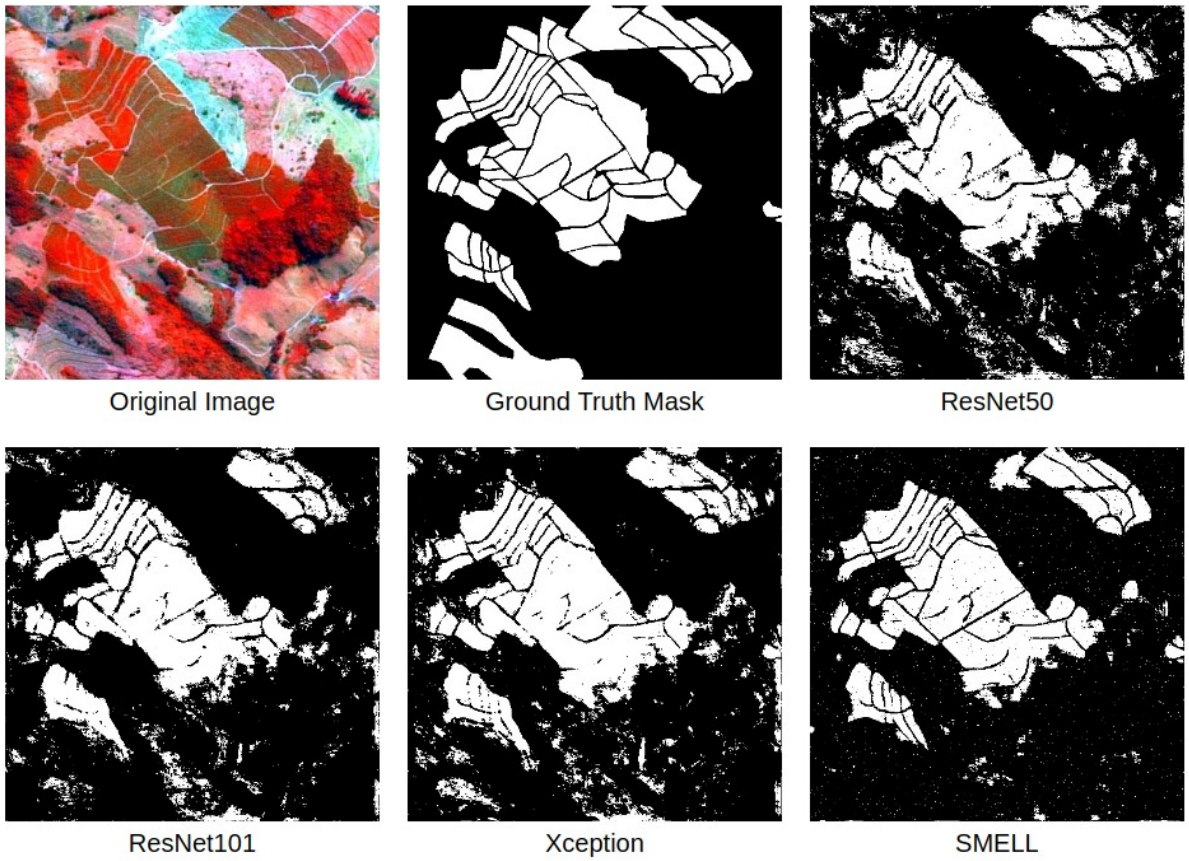

Figure 5.13: Brazilian Coffee Scenes dataset segmentation results for all networks with original image and ground truth mask for comparison.

SMELL segments coffee crops more accurately than all baseline networks, with greater accuracy and fewer false positive cases, i.e, no-coffee pixels misclassified as coffee. It is also possible to notice that our proposal detects the tangled lines amid crop fields more correctly than other architectures. Despite the good performance obtained, there are still some misclassified pixels spread around the generated mask. In the next subsection we present the results of SLIC application to the generated mask aiming to group crop field and no crop field areas to reduce the amount of such misclassifications cases.

5.3.1 Superpixel

In this subsection we explore SLIC superpixel technique aiming to improve the final accuracy of SMELL. The same range of parameters (number of segments and compactness) applied to the Inria dataset experiments were used to evaluate SLIC contribution to the prediction vector.

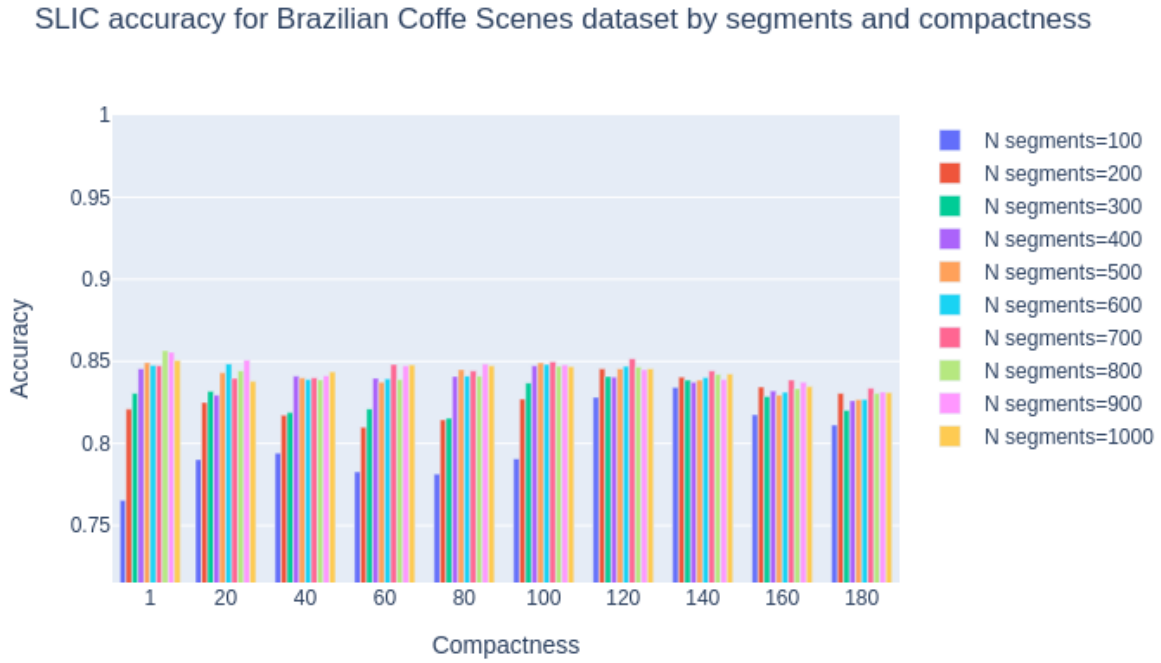

Figure 5.14: SLIC accuracy results for different number of segments - from 1 to 180 with a step of 20 - and compactness - from 100 to 1000 with a step of 100 for the Brazilian Coffee Scenes dataset.

Unlike building segmentation scenario, SLIC does not significantly enhance the final accuracy for the model for coffee crops segmentation. The best parameters combination found was compactness equals to 1 with 800 segments and it generates masks with almost the same accuracy as SMELL without SLIC - around 85%. The conclusion that SLIC technique does not improve the accuracy of SMELL shows that its applications as a post-processing method must be carefully evaluated on a case-by-case basis, as a bad combination of parameters can harm the final segmentation results.

Although SLIC application consolidates coffee and non coffee areas reclassifying pixels based on its neighborhood pixels, the accuracy for the tangled paths existing among coffee plantations suffers a notable decrease, as can be seen in Figure 5.15. This happens because in such cases the neighborhood of a path pixel is surrounded of coffee crop pixels, making it difficult to create a segment that groups the path pixels without transforming any coffee pixels into non-coffee pixel and vice-versa.

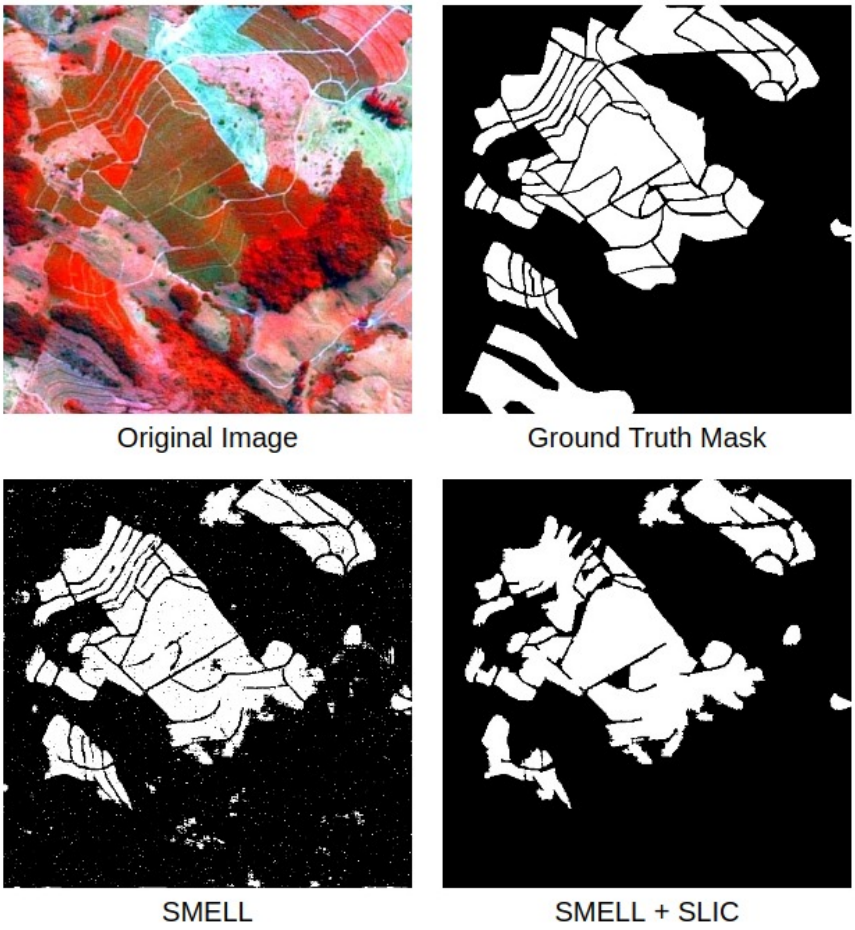

Figure 5.15: An example of an input image with (left top) its corresponding mask (right top) and the segmentation output from SMELL (left bottom) and from SMELL with SLIC (right bottom) for Brazilian Coffee Scenes dataset.

The generated masks shows that SMELL without SLIC is more precise overall, with paths amid coffee crops well defined and some isolated cases of misclassification, while the generated mask using SMELL with SLIC better defines macro regions of coffee crops, removing the spread misclassified cases and creating more homogeneous areas but impairing the correct classification of tangled paths. Also, differently than some results for Inria dataset, the agriculture segmentation task is not saturated and has room for improvement, presenting some areas of imprecise classification. For the example above, parts of coffee plantations that are light blue in color are almost entirely misclassified as non-coffee areas. At the same time, these areas have this hue due to the plantations not being grown, resembling areas with no coffee plantation. Thus, even missing the truth label for this cases, SMELL presents a good judgment of what is a coffee crop and what isn't.

5.4 Evaluation in tree and car segmentation

In this section we present the experiments related to Potsdam dataset. We select two classes from the original multi class masks in this dataset to test: car and tree. For each case, we tested a small and a large context window size (7×7 and 21×21) aiming to find the best set up for each segmentation scenario. The following table presents the results regards car segmentation.

Table 5.9: Potsdam car segmentation results for small and large context windows size with standard deviation for a 10-Fold Cross Validation.

Context Windows Size	Method	Accuracy	F1-Score	Precision
7 x 7	**ResNet50**	0.9947 (0.0013)	0.9921 (0.0019)	0.9895 (0.0026)
	ResNet101	0.9950 (0.0011)	0.9925 (0.0015)	0.9901 (0.0022)
	Xception	0.9952 (0.0020)	0.9928 (0.0016)	0.9904 (0.0019)
	SMELL	0.9924 (0.0029)	0.9937 (0.0015)	0.9959 (0.0009)
21 x 21	**ResNet50**	0.9952 (0.0020)	0.9928 (0.0016)	0.9904 (0.0023)
	ResNet101	0.9923 (0.0024)	0.9908 (0.0014)	0.9908 (0.0019)
	Xception	0.9934 (0.0022)	0.9915 (0.0013)	0.9920 (0.0011)
	SMELL	0.9921 (0.0019)	0.9920 (0.0009)	0.9924 (0.0003)

From the metrics obtained, it is evident that both context windows size presented similar results. Also, the results seems to be saturate, with all tested networks achieving more than 99% of accuracy regardless of the amount of contextual information provided. Although the metrics indicate success for the car segmentation task, the generated masks show the opposite. Figure 5.16 shows an example of car segmentation visual results and shows that the tested baselines networks tend to classify all pixels as non car, achieving an unrepresentative result for the original image. The reason for such behaviour is the fact that car masks for the Potsdam dataset have imbalanced labels, making possible to achieve high accuracy results by classifying all cases equally as the majority class. The results show that SMELL does not suffer with the imbalance dataset, being able to distinguish even the pixels of windows and car headlights, generating good visual results.

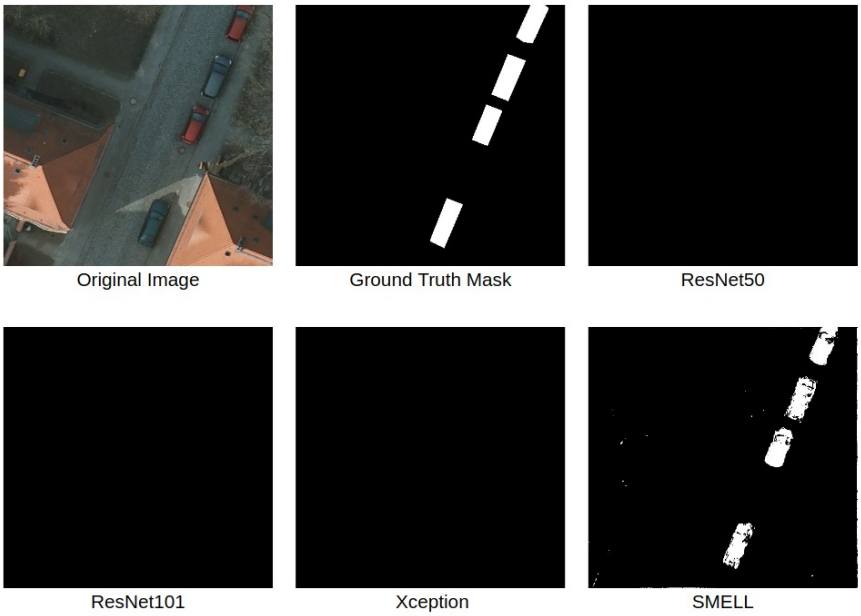

Figure 5.16: Potsdam dataset car segmentation task generated masks.

For the tree segmentation task, a different behaviour is observed. Different architectures better adapt to different sizes of context windows. A small context window size is more suitable for ResNet50 and ResNet101 while Xception and SMELL performs better with large context windows size. SMELL surpasses all networks and context windows combination with a 21×21 context window size in F1-Score metric. In addition, SMELL seems to be more consistent among different evaluation metrics, maintaining precision close to accuracy and F1-Score, which is not the case for Xception. The results for tree segmentation are displayed in the following table. Despite Xception achieve a better accuracy than SMELL, the visual results show that Xception suffers to correctly classify tree pixels, as in car segmentation scenario. SMELL generates the best visual results obtained through all tested baselines networks, as observed in Figure 5.17.

Table 5.10: Potsdam tree segmentation results for small and large context windows size with standard deviation for a 10-Fold Cross Validation.

Context Windows Size	Method	Accuracy	F1-Score	Precision
7 x 7	**ResNet50**	0.8922 (0.0244)	0.8695 (0.0084)	0.8498 (0.0094)
	ResNet101	0.8550 (0.0019)	0.7956 (0.0004)	0.7634 (0.0032)
	Xception	0.8604 (0.0184)	0.7958 (0.0172)	0.7403 (0.0087)
	SMELL	0.7161 (0.0282)	0.7724 (0.0211)	0.8953 (0.0098)
21 x 21	**ResNet50**	0.7578 (0.0451)	0.7459 (0.0297)	0.7449 (0.0176)
	ResNet101	0.7897 (0.0164)	0.7528 (0.0125)	0.7575 (0.0227)
	Xception	**0.9156 (0.0152)**	0.8753 (0.0045)	0.8383 (0.0056)
	SMELL	0.9072 (0.0050)	**0.9061 (0.0043)**	**0.9062 (0.0076)**

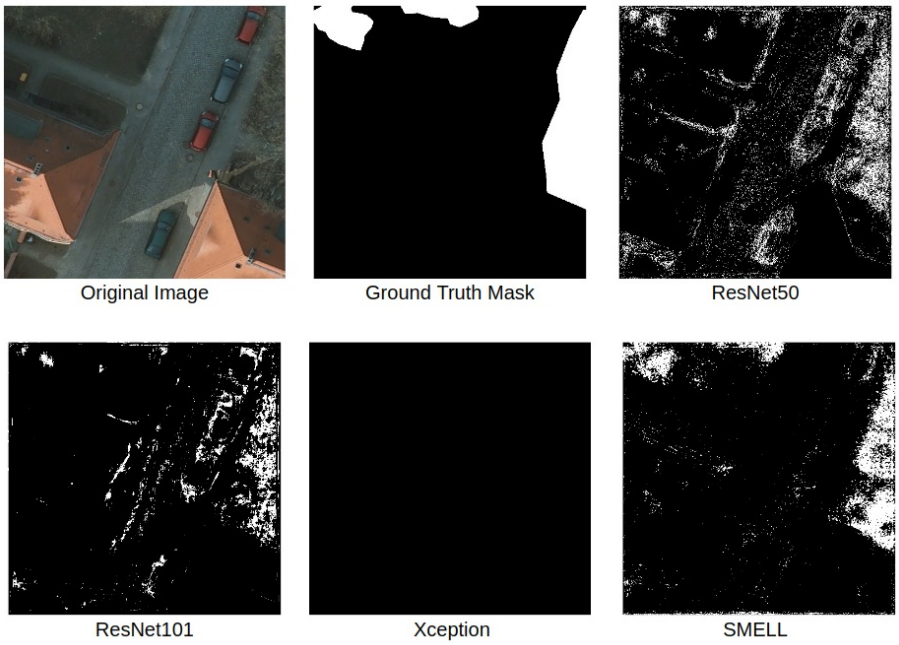

Figure 5.17: Potsdam dataset tree segmentation task generated masks.

5.4.1 Superpixel

Following the methodology applied to the other datasets, for the Potsdam dataset, we applied SLIC superpixel as post processing technique to improve SMELL's output mask. We tested the same range of parameters previously defined for building segmentation tasks, and evaluate each combination using accuracy as default metric. Figure 5.18 presents the accuracy for different combinations of compactness and number of segments. For 1000 segments and compactness equals to 20, SLIC improves SMELL's output mask accuracy by 3%.

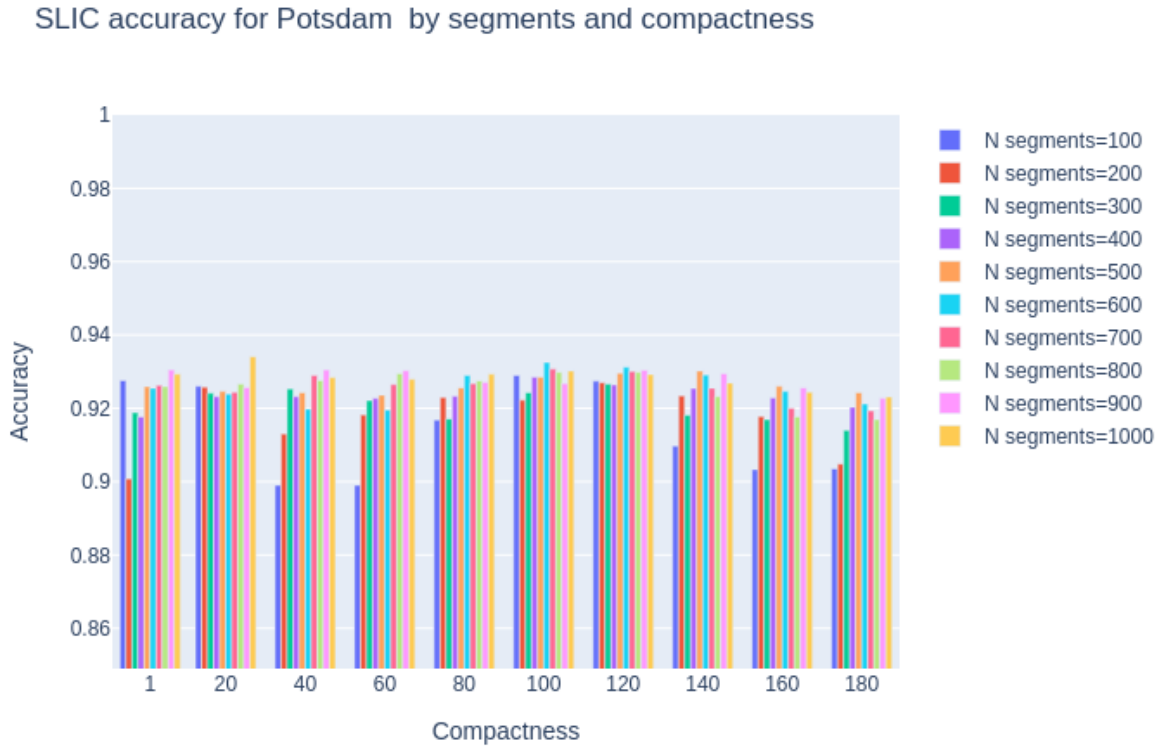

Figure 5.18: SLIC parameters combination for tree segmentation task with Potsdam dataset. The best combination of parameter is compactness equals to 20 and 1000 segments, achieving a 0.924 accuracy.

Similar to the results obtained for other datasets, SLIC increases the results metrics, better defining tree and non-tree areas based on the neighborhood pixels. For the car segmentation task, since the results are saturate with a high accuracy, we did not applied SLIC methodology. Figure 5.19 shows an example of SMELL's output with SLIC. Although SLIC seems to improve the result metrics, the mask output does not suffer a big improvement in tree classification. Positively, misclassified pixels spread through the image is fixed using SLIC. However, the tree areas are not precisely group together, which causes a single tree to be segmented as multiple objects of this class, sometimes giving a wrong idea of the positioning of trees in the image.

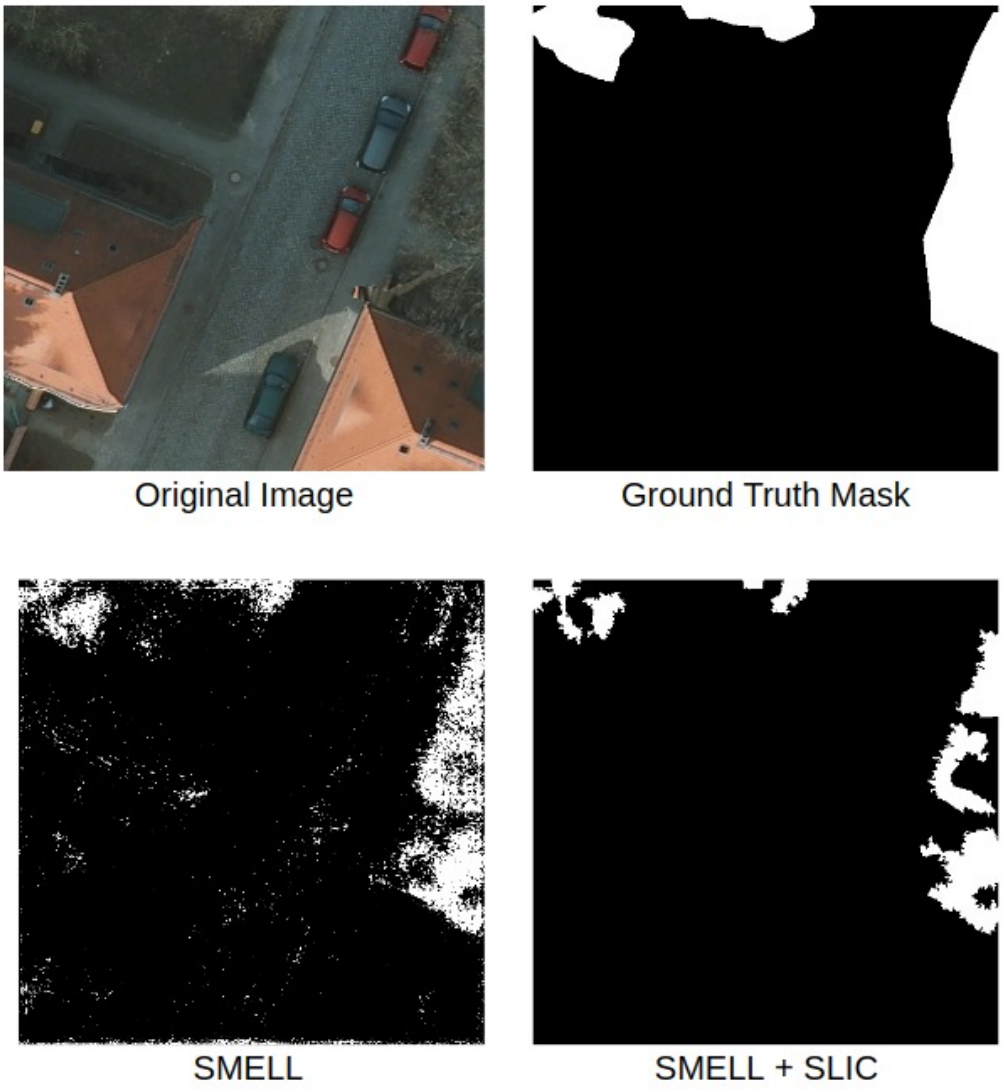

Figure 5.19: SLIC parameters combination for tree segmentation task with Potsdam dataset. The best combination of parameter is compactness equals to 20 and 100 segments, achieving a 0.924 accuracy.

Chapter 6

Conclusion

This work explored the possibility of a Deep Metric Learning algorithm for semantic segmentation in remote sensing images. We adapted a siamese autoencoder based structure with a metric learning module - namely SMELL [4] - originally designed for image classifications tasks using the context windows methodology proposed by [31]. We evaluate the effects of the use of different amounts of contextual information and the impact of SLIC superpixel as a post processing technique to generate more accurate predictions masks.

The proposed adaptation was tested in four different binary segmentation scenarios: building, coffee crop, car and tree. For the building segmentation task we first tested four different autoencoder architectures, aiming to find if more convolutional layers - and consequently more filters in the latent space - directly impacts the performance of SMELL. This test made it possible to conclude that a more deep and complex autoencoder structure and a more shallow version of it performs almost at the same level, leading us to choose the shallow version for faster training. Then we used five cities from the Inria dataset for experiments and we vary the context windows size from 15×15 to 21×21.

The results show that providing more contextual information improves network performance for all tested networks, reaching around 90% accuracy for the worst case (Chicago) and slightly outperforming the state-of-art architectures. For less populated cities the results appeared to be saturated, with accuracy greater than 99% for Kitsap. Finally, the application of SLIC superpixel technique as post-processing significantly improved the metrics for building segmentation as it present homogeneous spaces for building.

For the Brazilian Coffee scenes dataset, SMELL exceeded the baseline networks by about 3% across all metrics, yielding more accurate masks for coffee crop pixel wise classification task. We vary the context windows size from 17×17 to 23×23 with similar conclusions to building segmentation, i.e, more contextual information generates more accurate predictions. For the 23×23, due to a lack of computational memory, we trained SMELL using half the amount of images than used in the other sizes of context windows. Even with less images available to train, the performance metrics were greater for large contextual information, meaning that the contextual information can be considerable more important for pixel wise classification than the amount of data itself, when enough data is

provided to the model. In this experiments, SLIC did not improve SMELL's results due to heterogeneity of nearby areas, specially the roads (or paths) amid coffee plantations.

Finally for the Potsdam dataset, we tested two binary scenarios: car and tree segmentation. For the first scenario (car), the results showed saturation as for the metrics. However, because it is an imbalanced classes scenario, the tested baselines architectures presented output masks with all pixels classified as non-car for the chosen example image. On the other hand, SMELL presented a very precise segmentation mask, with cars pixels well defined. For the tree scenario, the obtained metrics were in general higher for SMELL than for the baselines networks. The application of the SLIC superpixel improved the metrics by about 3%, but generating more diffused tree-segmented areas.

Overall, the results prove that the use of a DMeL method for the semantic segmentation of aerial images can perform as well as, or even surpass, state-of-the-art neural network architectures. In addition, the adaptation of classification networks for semantic segmentation through the context windows methodology, despite generating a bottleneck in computational memory, which makes the training process very slow and limited in relation to the amount of data, proves to be very consistent even for tasks using high dimensional images.

6.1 Future Works

After a vast set of experiments with context windows as adaptation methodology from classification to semantic segmentation tasks of aerial images, an urge for possible improvements in our proposal arises. The memory amount needed for context windows application creates a bottleneck for high dimensional data. Changing the adaptation method of SMELL seems to be the first big step towards an end-to-end DMeL framework capable of dealing with high dimensional imagery. Since a variational autoencoder maps the data into a probability space, switching the convolutional autoencoder to a variational autoencoder may be a good start point to fully adapt SMELL to segmentation tasks. However, this type of modification directly impacts the S-space, increasing the number of points to be grouped around the markers, possibly generating another lack of memory.

Other interesting analysis for SMELL takes place in the S-space markers exploration. In this work, we focused on testing SMELL's performance and compare it against state-of-art models for many scenarios. A deeper dive into the markers and the similarity regions may bring interesting insights in order to perfect SMELL for segmentation tasks. Finally, the application in multi class scenarios is suggested. It would require the use of a more complex final predictor than OLS applied here for binary pixel wise classifications.

For this case, we started some experiments with a KNN predictor. However, a more complex predictor - such SVM - seems to aggravate the memory bottleneck for SMELL with context windows method. So, we suggest that the adaptation method undergoes the first changes and, when it is possible to train the network without generating one image for each pixel, SMELL can be tested in multi class segmentation scenarios.

Bibliography

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

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[3] Dana H. Ballard. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, page 279–284. AAAI Press, 1987.

[4] Pedro H. Barros, Fabiane Queiroz, Flavio Figueredo, Jefersson A. dos Santos, and Heitor S. Ramos. A new similarity space tailored for supervised deep metric learning, 2020.

[5] Luca Bertinetto, Jack Valmadre, Joao Henriques, Andrea Vedaldi, and Philip Torr. Fully-convolutional siamese networks for object tracking. volume 9914, pages 850–865, 10 2016.

[6] Jun Cen, Peng Yun, Junhao Cai, Michael Yu Wang, and Ming Liu. Deep metric learning for open world semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15333–15342, October 2021.

[7] S. Chopra, R. Hadsell, and Y LeCun. Learning a similarity metric discriminatively, with application to face verification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–546, 2005.

[8] Bin Deng, Sen Jia, and Daming Shi. Deep metric learning-based feature embedding for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 58(2):1422–1435, 2020.

[9] Rongsheng Dong, Lulu Bai, and Fengying Li. Siamesedenseu-net-based semantic segmentation of urban remote sensing images. *Mathematical Problems in Engineering*, 2020, 2020.

[10] Jefersson dos Santos, Philippe-Henri Gosselin, Sylvie Philipp-Foliguet, Ricardo Torres, and Alexandre Falcão. Multiscale classification of remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 50, 10 2012.

[11] Diego Fustes, Diego Cantorna, Carlos Dafonte, Bernardino Arcay, Alfonso Iglesias, and Minia Manteiga. A cloud-integrated web platform for marine monitoring using gis and remote sensing. application to oil spill detection through sar images. *Future Generation Computer Systems*, 34:155–160, 2014. Special Section: Distributed Solutions for Ubiquitous Computing and Ambient Intelligence.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[13] Zhiqiang Gong, Ping Zhong, Weidong Hu, and Yuming Hua. Joint learning of the center points and deep metrics for land-use classification in remote sensing. *Remote Sensing*, 11(1), 2019.

[14] Zhiqiang Gong, Ping Zhong, Yang Yu, and Weidong Hu. Diversity-promoting deep structural metric learning for remote sensing scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 56(1):371–390, 2018.

[15] S. Gould, J. Rodgers, and D. et al. Cohen. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 2008.

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

[17] Hecht-Nielsen. Theory of the backpropagation neural network. In *International 1989 Joint Conference on Neural Networks*, pages 593–605 vol.1, 1989.

[18] Sixing Hu, Mengdan Feng, Rang M. H. Nguyen, and Gim Hee Lee. Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7258–7267, 2018.

[19] Tomas Iesmantas, Agne Paulauskaite-Taraseviciene, and Kristina Sutiene. Enhancing multi-tissue and multi-scale cell nuclei segmentation with deep metric learning. *Applied Sciences*, 10(2), 2020.

[20] Jian Kang, Ruben Fernandez-Beltran, Zhirui Wang, Xian Sun, Jingen Ni, and Antonio Plaza. Rotation-invariant deep embedding for remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13, 2022.

[21] Jian Kang, Ruben Fernandez-Beltran, Zhen Ye, Xiaohua Tong, Pedram Ghamisi, and Antonio Plaza. Deep metric learning based on scalable neighborhood components for remote sensing scene characterization. *IEEE Transactions on Geoscience and Remote Sensing*, 58(12):8905–8918, 2020.

[22] Mahmut Kaya and H. Bilge. Deep metric learning: A survey. *Symmetry*, 11:1066, 08 2019.

[23] Xinhua Liu, Gaoqiang Hu, Xiaolin Ma, and Hailan Kuang. An enhanced neural network based on deep metric learning for skin lesion segmentation. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 1633–1638, 2019.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[25] Jiwen Lu, Junlin Hu, and Jie Zhou. Deep metric learning for visual understanding: An overview of recent advances. *IEEE Signal Processing Magazine*, 34(6):76–84, 2017.

[26] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2017.

[27] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.

[28] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[29] Wojciech Mokrzycki and Maciej Tatol. Color difference delta e - a survey. *Machine Graphics and Vision*, 20:383–411, 04 2011.

[30] Lichao Mou, Yuansheng Hua, and Xiao Xiang Zhu. A relation-augmented fully convolutional network for semantic segmentation in aerial scenes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] Keiller Nogueira, Mauro Dalla Mura, Jocelyn Chanussot, William Robson Schwartz, and Jefersson A. dos Santos. Learning to semantically segment high-resolution remote sensing images. *23rd International Conference on Pattern Recognition (ICPR)*, 2016.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[33] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[34] Qian Shi, Mengxi Liu, Shengchen Li, Xiaoping Liu, Fei Wang, and Liangpei Zhang. A deeply supervised attention metric-based network and an open aerial image dataset for remote sensing change detection. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–16, 2022.

[35] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012, 2016.

[36] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012, 2016.

[37] Yu Sun, Fukun Bi, Yangte Gao, Liang Chen, and Suting Feng. A multi-attention unet for semantic segmentation in remote sensing images. *Symmetry*, 14(5), 2022.

[38] Michele Volpi and Vittorio Ferrari. Semantic segmentation of urban scenes by learning local class interactions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–9, 2015.

[39] Wanyin Wu, Dapeng Tao, Hao Li, Zhao Yang, and Jun Cheng. Deep features for person re-identification on metric learning. *Pattern Recognition*, 110:107424, 2021.

[40] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[41] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Deep metric learning for person re-identification. In *2014 22nd International Conference on Pattern Recognition*, pages 34–39, 2014.

[42] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Deep metric learning for person re-identification. In *2014 22nd International Conference on Pattern Recognition*, pages 34–39, 2014.

[43] Min-Sub Yun, Woo-Jeoung Nam, and Seong-Whan Lee. Coarse-to-fine deep metric learning for remote sensing image retrieval. *Remote Sensing*, 12(2), 2020.

[44] Molan Zhang and ZhiQiang Chen. Deep metric learning for damage detection using bitemporal satellite images. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 562–565, 2021.

[45] Hongwei Zhao, Lin Yuan, Haoyu Zhao, and Zhen Wang. Global-aware ranking deep metric learning for remote sensing image retrieval. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.

[46] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE Transactions on Medical Imaging*, 39:1856–1867, 2020.