

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

Pedro Henrique Targino Gama

Meta learning Approaches for Few-Shot Semantic Segmentation with Sparse  
Labels

Belo Horizonte  
2021

Pedro Henrique Targino Gama

**Meta learning Approaches for Few-Shot Semantic Segmentation with Sparse  
Labels**

**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  
Co-Advisor: Hugo Neves de Oliveira

Belo Horizonte  
2021



© 2021, Pedro Henrique Targino Gama.  
. Todos os direitos reservados

Gama, Pedro Henrique Targino.

G184m Meta-learning approaches for few-shot semantic segmentation with sparse labels [manuscrito] / Pedro Henrique Targino Gama. — Belo Horizonte, 2021.  
95 f. il.; 29 cm.

Orientador: Jefersson Alex dos Santos.  
Coorientador: Hugo Neves de Oliveira  
Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação  
Referências: f.81-88.

1. Computação – Teses. 2. Visão por computador – Teses. 3. Aprendizagem de máquina – Teses. 4. Aprendizado profundo – Teses. 5. Meta-aprendizado – Teses. I. Santos, Jefersson Alex dos. II. Oliveira, Hugo Neves de. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. IV. Título.

CDU 519.6\*82.10 (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  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

## FOLHA DE APROVAÇÃO


Meta learning Approaches for Few-Shot Semantic Segmentation with  
Sparse Labels


**PEDRO HENRIQUE TARGINO GAMA**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

  
PROF. JEFERSSON ALEX DOS SANTOS - Orientador  
Departamento de Ciência da Computação - UFMG

  
DOUTOR HUGO NEVES DE OLIVEIRA - Coorientador  
Instituto de Matemática e Estatística - USP

  
PROF. ADRIANO ALONSO VELOSO  
Departamento de Ciência da Computação - UFMG

  
PROF. GILSON ALEXANDRE OSTWALD PEDRO COSTA  
Instituto de Matemática e Estatística - UERJ

Belo Horizonte, 15 de Abril de 2021.

*Dedico esse trabalho as mais de 600 mil vítimas de COVID-19 no Brasil e a todos que sofrem com essa tragédia, que poderia ser mitigada por autoridades competentes que se preocupassem com a população.*

# Acknowledgments

Gostaria de primeiramente agradecer aos meus pais. O suporte deles é o que me garantiu a estabilidade que preciso para me dedicar a minha vida acadêmica e a esse trabalho.

Também agradeço aos meus amigos que fiz nessa jornada pela UFMG. Obrigado pelas risadas e horas de entretenimento, e pelo apoio e encorajamento. Principalmente nesse período de quarentena, vocês foram essenciais para me manter saudável. Aos amigos do PATREO, agradeço também por estarem sempre abertos para ajudar, seja discutindo soluções, abordagens diferentes, ou resolvendo *aqueles* bugs. Em especial, gostaria de agradecer aos meus dois amigos, Pedro Otávio e Lucas Magno, que me aturam desde o Ensino Médio. A companhia de vocês é muito importante, e sempre estavam dispostos a me dar opiniões construtivas (mesmo que não entendessem sobre o assunto).

Por fim, gostaria de agradecer aos meus orientadores Jefersson e Hugo, além de muitos dos professores que tive contato, pelos ensinamentos e dicas ao longo dos anos. Jefersson, obrigado pela liberdade e confiança. Hugo sem você esse trabalho não teria saído do papel.

*“True, we don’t have anything such as ‘fate’.  
It’s only those who drink in ignorance and fear  
and stumble over their own feet  
that fall and disappear within the muddy river  
known as ‘fate’.”*

(Tite Kubo, Bleach Volume 6)

# Resumo

Segmentação Semântica é uma tarefa clássica de visão computacional que tem múltiplas aplicações em diversas áreas, desde de segmentação de órgãos para estudos clínicos por imagem, contagem de objetos em linha de produção, até a estimativa de tamanho de áreas de desmatamento. Entretanto, o tipo de rotulação de dados necessária para os métodos atuais resolverem o problema é laboriosa de se produzir, uma vez que é necessário determinar os rótulos de todos os pixels da imagem. Isso costuma aumentar o custo (humano e/ou monetário) de construção de novos conjuntos de dados. Duas formas possíveis de se reduzir esse custo são: 1) diminuindo o número de imagens anotadas; 2) usando um formato de anotação mais simples/esparsa. Porém, os métodos comuns e mais atuais, de deep learning, para segmentação semântica não funcionam bem usando uma, ou duas, dessas soluções. Neste trabalho propomos dois métodos de meta learning para segmentação semântica em cenários few-shot com rotulação esparsa. Essas abordagens foram baseadas em dois métodos existentes para classificação: *Model-Agnostic Meta-Learning* (MAML) e *Prototypical Networks*. As nossas abordagens foram testadas em diversos cenários da área médica e sensoriamento remoto, que normalmente tem uma limitação de aquisição de dados, e obtiveram resultados competitivos em diferentes tarefas.

**Palavras-chave:** Computação, Visão Computacional, Aprendizado de Máquina, Aprendizado Profundo, Meta-Aprendizado.

# Abstract

Semantic Segmentation is a classic task in Computer Vision that has multiple applications in many areas, from organ segmentation for clinical image studies, or counting objects in production lines, to estimating deforestation areas sizes. However, the type of data labeling required for actual methods to solve this problem is laborious to produce, since one has to determine the label for all pixels of an image. This usually increases the cost (human and/or monetary) to produce new datasets. Two possible ways to reduce this cost are: 1) reducing the number of labeled samples; 2) using simpler/sparse types of annotation. Despite that, current and usual deep learning based methods for segmentation tend to perform poorly when using one, or two, of these solutions. In this work, we propose two meta learning methods to the few-shot semantic segmentation task with sparse annotations. These two approaches are based on two existing methods for classification: Model-Agnostic Meta-Learning (MAML) and Prototypical Networks. Our methods were tested in different scenarios in the medical and remote sensing areas, which usually have limited data access, and obtained competitive results in different tasks.

**Keywords:** Computing, Computer Vision, Machine Learning, Deep Learning, Meta Learning.

# List of Figures

1.1	Example of semantic segmentation. The image of the cat in the left is segmented, creating 3 semantic groups: cat, box, and background. . . . .	16
1.2	Examples of different types of sparse annotations. In the upper row, the image and original dense annotation. On the bottom, from left to right, there are an example of points, scribbles, and contour annotations. . . . .	18
2.1	Illustration of a Convolutional operation in a 2D matrix. . . . .	23
2.2	Illustration of a feature map. The filter is applied at all valid locations of the input producing the feature map. . . . .	24
2.3	The increased <i>receptive field</i> of a filter after a pooling operation. . . . .	25
2.4	Illustration of a simple CNN architecture. . . . .	25
2.5	Illustration of the original UNet architecture. (Source: <a href="#">Ronneberger et al. [2015]</a> )	26
2.6	MAML Diagram. The parameter $\theta$ is optimized by training on multiple tasks $\mathcal{T}_i$ , by following their losses $\mathcal{L}_i$ , and thus the optimized $\theta^*$ can be fastly adapted to a new task $\mathcal{T}_K$ . . . . .	27
2.7	Illustration of the Prototypical Networks. The embedding function $f_\Phi$ embed the support set, which is used to compute the class prototypes $\mathbf{c}_k$ . The embedded query image is compared to the class prototypes to predict its class.	30
4.1	Illustration of the WeaSeL method with toy examples. . . . .	39
4.2	Illustration of a Masked Average Pooling. Our process differ after masking the features. A common Masked Average Pooling, will average these features to each sample, and then average these averages (right). Ours will create the <i>global sample average</i> (bottom) by considering all pixels. . . . .	41
5.1	Examples from the Chest X-Ray Datasets. . . . .	46
5.2	Examples from the Mammography Datasets. . . . .	46
5.3	Examples from the Dental X-Rays Datasets. . . . .	47
5.4	Examples from the Brazilian Coffee and Orange Orchards Dataset. . . . .	49
5.5	Examples from the ISPRS Datasets. . . . .	50
5.6	Examples from the INRIA Dataset. . . . .	51
5.7	Illustration of the types of sparse annotations used. Annotations are illustrative and uspcaled for better visualization. . . . .	52
5.8	Illustration of the miniUNet architecture. The upper numbers represents the feature dimension of the volumes, while on the side is the spatial dimensions. .	55



6.1	Jaccard Score of experiments with JSRT Lungs Task. . . . .	60
6.2	Jaccard Score of experiments with JSRT Heart Task. . . . .	61
6.3	Jaccard Score of experiments with OpenIST Lungs Task. . . . .	61
6.4	Jaccard Score of experiments with Montgomery Lungs Task. . . . .	62
6.5	Jaccard Score of experiments with NIH-Labeled Lungs Task. . . . .	62
6.6	Jaccard Score of experiments with MIAS Breast Task. . . . .	63
6.7	Jaccard Score of experiments with INbreast Pectoral Muscle Task. . . . .	64
6.8	Jaccard Score of experiments with Panoramic Mandible Task. . . . .	64
6.9	Jaccard Score of experiments with UFBA-UESC Teeth Task. . . . .	65
6.10	Jaccard Score of experiments with Montesanto Coffee Task. . . . .	66
6.11	Jaccard Score of experiments with Arceburgo Coffee Task. . . . .	67
6.12	Jaccard Score of experiments with Orange Orchard Task. . . . .	67
6.13	Jaccard Score of experiments with Austin Building Task. . . . .	68
6.14	Jaccard Score of experiments with Kitsap County Building Task. . . . .	69
6.15	Jaccard Score of experiments with Vienna Building Task. . . . .	69
6.16	Jaccard Score of experiments with Vaihingen Roads Task. . . . .	71
6.17	Jaccard Score of experiments with Vaihingen Tree Task. . . . .	71
6.18	Jaccard Score of experiments with Potsdam Buildings Task. . . . .	72
6.19	Jaccard Score of experiments with Potsdam Roads Task. . . . .	72
6.20	Number of user Inputs versus Jaccard Score in the JSRT Lungs Task. . . . .	73
6.21	Number of user Inputs versus Jaccard Score in the Montesanto Coffee Task. . . . .	73
6.22	Number of user Inputs versus Jaccard Score in the Austin Buildings Task. . . . .	74
6.23	Number of user Inputs versus Jaccard Score in the Vaihingen Buildings Task. . . . .	74
6.24	Jaccard Score by few-shot fine-tuning Epoch in the OpenIST Lungs task using Contours annotation. . . . .	76
6.25	Jaccard Score by few-shot fine-tuning Epoch in the INbreast Breast task using Points annotation. . . . .	77
6.26	Jaccard Score by few-shot fine-tuning Epoch in the Montesanto Coffee task using Skeletons annotation. . . . .	77
6.27	Jaccard Score by few-shot fine-tuning Epoch in the Arceburgo Coffee task using Regions annotation. . . . .	78
6.28	Jaccard Score by few-shot fine-tuning Epoch in the Orange Orchard task using Points annotation. . . . .	78
A.1	Jaccard Score of experiments with JSRT Clavicles Task. . . . .	90
A.2	Jaccard Score of experiments with Shenzhen Lungs Task. . . . .	91
A.3	Jaccard Score of experiments with MIAS Pectoral Muscle Task. . . . .	91
A.4	Jaccard Score of experiments with INbreast Breast Task. . . . .	92
A.5	Jaccard Score of experiments with Brazilian Coffee: Guaxupé Coffee Task. . . . .	92

A.6	Jaccard Score of experiments with Brazilian Coffee: Guaranésia Coffee Task. . .	93
A.7	Jaccard Score of experiments with INRIA: Chicago Building Task. . . . .	93
A.8	Jaccard Score of experiments with INRIA: Western Tyrol Building Task. . . .	94
A.9	Jaccard Score of experiments with ISPRS: Vaihingen Building Task. . . . .	94
A.10	Jaccard Score of experiments with ISPRS: Vaihingen Grass Task. . . . .	95
A.11	Jaccard Score of experiments with ISPRS: Potsdam Grass Task. . . . .	95
A.12	Jaccard Score of experiments with ISPRS: Potsdam Tree Task. . . . .	96

# List of Tables

3.1	Presented Literature Works . . . . .	36
5.1	Descriptions of the miniUNet Blocks . . . . .	54
5.2	Number of epochs for training the methods in different experiments. <i>#Epochs</i> <i>Tuning</i> is the number of epochs trained with the Few-Shot Task samples. . . .	57

# List of Algorithms

1	Model-Agnostic Meta Learning: Supervised version . . . . .	28
2	Prototypical Networks: Few-shot segmentation training . . . . .	42

# Contents

<b>1</b>	<b>Introduction</b>	<b>16</b>
1.1	Objectives, Research Questions, and Contributions . . . . .	19
1.2	Outline . . . . .	20
<b>2</b>	<b>Background Concepts</b>	<b>21</b>
2.1	Semantic Segmentation and Convolutional Neural Networks . . . . .	21
2.1.1	Convolutional Neural Networks . . . . .	22
2.1.2	Fully Convolutional Networks and UNet . . . . .	25
2.2	Model-Agnostic Meta-Learning (MAML) . . . . .	27
2.3	Prototypical Networks . . . . .	29
<b>3</b>	<b>Literature Review</b>	<b>31</b>
3.1	Weakly Supervised/Sparse Label Semantic Segmentation . . . . .	31
3.2	Few-shot Classification and Meta-learning . . . . .	32
3.3	Few-Shot Semantic Segmentation . . . . .	35
<b>4</b>	<b>Methodology</b>	<b>37</b>
4.1	Problem Definition . . . . .	37
4.2	Gradient-based Semantic Segmentation from Sparse Labels . . . . .	38
4.3	Prototypical Seeds for Sparse Segmentation . . . . .	40
<b>5</b>	<b>Experimental Setup</b>	<b>43</b>
5.1	Datasets . . . . .	43
5.1.1	Medical Datasets . . . . .	44
5.1.2	Remote Sensing Datasets . . . . .	47
5.2	Types of sparse annotation . . . . .	52
5.3	miniUNet architecture . . . . .	54
5.4	Evaluation Protocol . . . . .	56
5.4.1	Baselines . . . . .	56
5.4.2	Protocol and Metrics . . . . .	56
<b>6</b>	<b>Results and Discussion</b>	<b>59</b>
6.1	Few-shot Semantic Segmentation: Sparse vs Dense labels . . . . .	59
6.1.1	Medical Tasks . . . . .	60
6.1.1.1	Chest X-Ray Datasets . . . . .	60

6.1.1.2	Mammography X-Ray Datasets . . . . .	63
6.1.1.3	Dental X-Ray Datasets . . . . .	64
6.1.2	Agricultural Tasks . . . . .	65
6.1.3	INRIA Tasks . . . . .	66
6.1.4	ISPRS Tasks . . . . .	70
6.2	Sparse labels comparison . . . . .	70
6.3	WeaSeL Fast Adaptation . . . . .	75
<b>7</b>	<b>Conclusion and Future Works</b>	<b>79</b>
	<b>Bibliography</b>	<b>82</b>
	<b>Appendix A Additional results for the Section 6.1</b>	<b>90</b>
A.1	Extra Medical Tasks . . . . .	90
A.2	Extra Remote Sensing Tasks . . . . .	91

# Chapter 1

## Introduction

The semantic segmentation problem consists of defining regions in an image that belongs to the same object class, thus producing fine boundaries to objects in that image. That is, given an image, assign to every pixel a label that represents the object which the pixel belongs to (see Figure 1.1, for an example). The semantic segmentation task have been studied for a long time and has appeared in works as early as 1978 [Ohta et al., 1978].



Figure 1.1: Example of semantic segmentation. The image of the cat in the left is segmented, creating 3 semantic groups: cat, box, and background.

This task appears in different scenarios. In medicine, there are several objectives that use semantic segmentation, and some of them include: study of anatomical structures [Sharma et al., 2008; Tesař et al., 2008]; identify regions of interest, i.e., tumors, lesions, etc. [Moon et al., 2002; Prasad et al., 2008]; measure tissue volume to analyze a tumor growth [Pham et al., 1997]; and other applications. Another scenario that have many segmentation applications is remote sensing. Common problems that use semantic segmentation include: land use cover [Lobo, 1997; Huang et al., 2002; Lizarazo and Barros, 2010; Montoya-Zegarra et al., 2015; Kampffmeyer et al., 2016; Kaiser et al., 2017]; environmental disasters [Kataoka et al., 2016; Lopez-Fuentes et al., 2017]; and others [Nogueira et al., 2015; Audebert et al., 2017; Cheng et al., 2017]. Some works have tackled semantic segmentation in the video domain, for example [Shelhamer et al., 2016; Caelles et al., 2017]. Other works, such as [Paszke et al., 2016; Treml et al., 2016], focus on reducing the inference time required to segment the images. These optimizations can lead up to apply this models in embarked systems, such as autonomous driving cars.

---

Over the years, different works investigated the problem of semantic segmentation, and in the last few years the state-of-the-art have been dominated by Deep Learning (DL) methods. The most prominent of these methods is the Convolutional Neural Networks (CNNs) firstly proposed by LeCun et al. [1990], and their subsequent modifications proposed by different authors. In [Krizhevsky et al., 2012], the authors propose a CNN with some minor modifications. The use of Graphical Processing Units (GPUs) allowed the training of a large complex model (for the time), which achieved one of the best results in ILSVRC-2010<sup>1</sup> contest reaching a milestone in image classification. This result affected the Computer Vision area as a whole, and in conjunction with the crescent advances in hardware, rapidly made deep learning models the state-of-the-art in the subsequently years.

Although deep learning models proving to be the most apt to many computer vision problems, these models bring their own issues, especially during training. Machine Learning (ML) models require data to be optimized, and, in general, with more data, smaller errors are obtained when testing the models. However, deep learning models, which are usually larger than standard ML, can suffer from small amounts of data in training. There are two common problems that appear in such scenarios: *underfitting* and *overfitting* [Goodfellow et al., 2016, Chapter 5]. The underfitting occurs when the small amount of data are not sufficient to provide a low error rate in training, for example, because the model could not learn key structures of the input data. Meanwhile, overfitting happens when the model performs poorly on the test set, while having great performance in the training set. Given the large *capacity* [Goodfellow et al., 2016, Chapter 5] of deep learning models, they can practically memorize the small sample of training, and then not generalize to the test samples.

The data used to train ML models can be raw observations like images, audio records, text, etc., or they can include labels to the data points. These labels are specific for the task, for example, in classification one needs the classes in each data point. When the model is trained solely with labeled data it is called *Supervised Learning*, in other hand, when no labels are provided it is named *Unsupervised Learning*; The *Semi-supervised* scenario is when one uses both labeled and unlabeled data for training. For semantic segmentation, the labels consist of an annotation with the real class for all the pixels in that image (we call this annotation, *full* or *dense annotation*). This type of label is laborious to produce and usually expensive. Specific scenarios, like medicine or remote sensing, usually require specialists to annotate the data. Thus *sparse annotations* becomes an interesting solution. These types of annotation consist in only presenting a label to a small set of pixels of the image. Different types of pixel sets can be used, i.e., points, regions, line segments, and others (see Figure 1.2). This type of annotation reduces the time required to produce the labels for an image, but can be challenging to train a model

---

<sup>1</sup><http://www.image-net.org/challenges/LSVRC/2010/>



with, since limits the amount of information available to the model. Works like [Lin et al., 2016; Vernaza and Chandraker, 2017; Wang et al., 2018a] have used sparse labels for semantic segmentation with relative success. The use of incomplete information in the labels (i.e. using sparse annotations) during the training is commonly referred to as *Weakly Supervised Learning*.

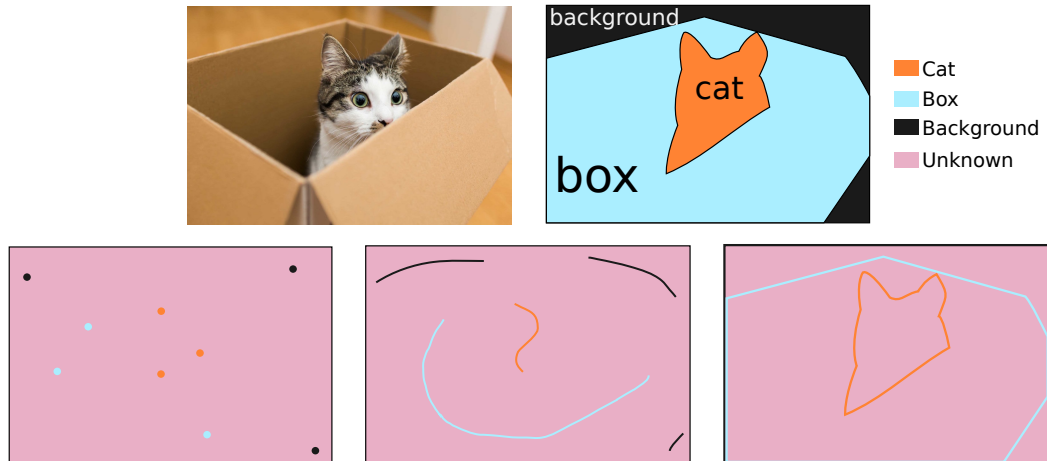


Figure 1.2: Examples of different types of sparse annotations. In the upper row, the image and original dense annotation. On the bottom, from left to right, there are an example of points, scribbles, and contour annotations.

Another strategy to reduce the cost of labeling datasets is to reduce the total number of images that composed the dataset, i.e. the number of labeled images. These scenarios with small number of image samples (less than 50, usually) are commonly known as Few-Shot, and recently have gained interest of the community. Few-Shot Learning have a large amount of works focused on the classification problem [Qiao et al., 2018; Vinyals et al., 2016; Snell et al., 2017; Simon et al., 2020; Finn et al., 2017; Raghu et al., 2019; Finn et al., 2018], although some works for semantic segmentation have been done in recent years [Shaban et al., 2017; Dong and Xing, 2018a; Hu et al., 2019; Zhang et al., 2020; Wang et al., 2019; Rakelly et al., 2018]. One methodology that has been successfully applied to few-shot problems in recent years is meta-learning. Normally understood as *learning to learn*, meta-learning is an umbrella term to a collection of methods that improves an learning algorithm through multiple learning episodes. In the survey Hospedales et al. [2020], the authors proposed a formalization of meta-learning and different forms to categorize methods that use this approach. We can summarize meta-learning methods in, an algorithm that learns a set of parameters  $\omega$  called *meta-knowledge*, trained in a distribution of tasks (a dataset and a loss function), such that  $\omega$  generalizes well for the tasks in said distribution. What comprises the *meta-knowledge* is defined by each method, and the authors of the survey use these assumptions to group meta learning methods. In this work, we will use adaptations of two methods of different categories, these methods being, Model-Agnostic Meta-Learning [Finn et al., 2017] and Prototypical Networks [Snell

et al., 2017], to the problem of semantic segmentation.

In this dissertation we want to investigate the feasibility of the use of sparse annotation to the few-shot semantic segmentation problem, in scenarios where data availability is limited, such as medical imaging and remote sensing. To tackle this problem, we will use adapted meta-learning methods that already been proved to be suitable to few-shot classification. We will show that our approaches can obtain results comparable to full annotations, using a variety of sparse annotations, for different tasks in medical and remote sensing scenarios.

## 1.1 Objectives, Research Questions, and Contributions

This work aims to study and find viable solutions to the problem of few-shot semantic segmentation with sparse annotations. Few-shot is already a problem that has a great limitation in data for the models, introducing sparse annotations will further increase this limitation and can hinder the process of training deep learn models. In order to direct our investigations we propose three questions to guide us, that are:

1. What is the impact of introducing sparse annotations in few-shot scenarios? Can the methods obtain results comparable to training with dense annotations?
2. What is the efficiency of different types of sparse annotation? How much data the user have to provide to obtain a useful model?
3. How different methods converge training in a few-shot task? The MAML algorithm is expected to produce a model able to *fast adapt*. This will occur in our test cases?

We designed our experiments to answer these questions and the results were promising. As the main contributions of this work, we present them as follow:

- The proposal of two meta-learning approaches for the problem of few-shot semantic segmentation with sparse labels, the *WeaSeL* and *Prototype* methods.
- An extensive evaluation of our proposals in a large set of tasks from Medical and Remote Sensing scenarios, using different task configurations.
- An analysis of five styles of sparse annotations, named: Points, Grid, Contours, Skeletons, and Regions.

---

## 1.2 Outline

The remainder of this work is organized as follows. In Chapter 2, we present some background concepts required for the understanding of this work, with a brief explanation of CNNs, semantic segmentation, and the two original methods for few-shot classification used as bases for our approaches. In Chapter 3, we present some of the related works in the literature, reviewing proposed methods for sparse labeled semantic segmentation, few-shot classification, and few-shot semantic segmentation. In Chapter 4, we present a definition for the few-shot semantic segmentation with sparse annotation and our two proposed approaches. The experimental setup is presented in Chapter 5, which includes a summary of the datasets used, the metrics, protocols, and baselines of our experiments. In Chapter 6, we show the results of our experiments and a brief discussion of them. Finally, in Chapter 7, we present our final remarks and future works.

# Chapter 2

## Background Concepts

In this chapter, we present and describe techniques and concepts relevant to the proposed work. First, in Section 2.1, we introduce some concepts of Convolutional Neural Networks and their usage in Semantic Segmentation with Fully Convolutional Networks. In Section 2.2, we present the MAML algorithm, while in Section 2.3 we present the Prototypical Networks method for the classification problem.

### 2.1 Semantic Segmentation and Convolutional Neural Networks

Early approaches for the problem of semantic segmentation could be, generally, divided in three steps. These steps can be organized in the pipeline below:

1. Pre-processing and Feature Extraction → 2. Classifier method → 3. Post-processing

In the first step, usually, some form of pre-processing is used to extract patches from the image. This pre-processing can be as simple as extracting patches from a grid, or more complex using some segmentation method such as Mean-Shift (MS) [Comaniciu and Meer, 2002] or the graph-cut based method of Felzenszwalb and Huttenlocher [2004]. Then, features are computed from the patches or raw pixels. These features were usually handcrafted or based in raw information such as values of pixel color. They could also be computed using other methods such as SIFT [Lowe et al., 1999], Fischer Kernels (FK) [Jaakkola and Haussler, 1999], or Bag of Visual Words (BOVW) [Csurka et al., 2004]. Using the extracted features, in the second step, classifier models are trained, and then assign class labels to the patches/pixels. Finally, a post-process is applied to rectify the labels assigned through some form of patch consistency, neighbor disparity, or other method.

Most of the works proposed over the years for semantic segmentation would be differentiated by which method used in each step of this pipeline, with some even blending some of the steps. [Csurka and Perronnin \[2008\]](#) and [Plath et al. \[2009\]](#) are examples of approaches that followed these steps.

One of the main drawbacks of this classical pipeline is that it was handcrafted for each problem. The method to extract features has to be selected, or specific ones have to be created. Also, the choice of the classifier is important and its efficiency can be affected by the features used. Moreover, even if effective for one problem, the features and classifiers often had to be changed for a new/different scenario.

### 2.1.1 Convolutional Neural Networks

The Convolutional Neural Network proposed by [LeCun et al. \[1990\]](#) is a Neural Network (NN) model that joins the firsts two steps of the pipeline above, that is the *Feature Extraction* and *Classifier method*, in a single entity. This neural network uses *convolutional layers* in its architecture, hence its name. It regained attention in the Computer Vision community after the work of [Krizhevsky et al. \[2012\]](#). This work includes adjustments to the architecture, a larger number of layers and changing their configurations, the use of the ReLU activation function (equation 2.3), but most important was the use of GPUs to train their CNN model that was used to solve the large classification problem presented in the ImageNet LSVRC-2010 contest<sup>1</sup>.

A convolutional layer is composed of a set of convolutional filters (also called kernels). A filter is a parametrized matrix, or tensor, that is used in a convolutional operation over a volume (i.e. an image or the output of another layer). The convolutional operation is a mathematical operation over two functions  $f$  and  $g$  and is defined by:

$$(f * g)(x) := \int_{-\infty}^{\infty} f(t)g(x - t)dt \quad (2.1)$$

However, in CNNs we use a discrete version of the operation. In fact, the accurate operation used is the discrete Cross-Correlation, but because the similarity of these operations and the use of, mostly, real valued variables, the community conventionally named it convolution. Thus, for clarity, we will define the a convolution operation in a CNN.

---

<sup>1</sup>ImageNet 2010: <http://www.image-net.org/challenges/LSVRC/2010/>

Given an input  $f \in \mathbb{R}^{N \times N}$  and a filter  $g \in \mathbb{R}^{M \times M}$ , we define the CNN convolution of  $(f * g)$  applied at the point  $(i, j)$  as follow:

$$(f * g)[(i, j)] = \sum_{r=0}^M \sum_{c=0}^M f[i-r, j-c]g[r, c] \quad (2.2)$$

If the input  $f \in \mathbb{R}^{N \times N \times D}$ , how the convolution operation is applied over the additional dimension (often referred to as *feature dimension*) will depend on the CNN architecture. Some architectures implement the use of one filter that is applied to each feature plane ( $f^{N \times N \times d}$ , for some fixed  $d$ ) and then sum over the filters to produce a single output. Others apply a single 3D filter to the input volume. Different variations of grouping filters can also be used. Figure 2.1 illustrate the convolutional operation of a CNN with 2D filter and input. The result of the application of a filter in a specific point can be interpreted as a single neuron in that layer, that have access only to the data in the window, that is defined by the filter size, around the application point. This area is usually named the *receptive field* of the neuron. A convolutional filter will be applied in all valid locations of the input producing a matrix output were each position is the result of the convolution of the filter on a respective position in the input. Depending on the kernel size, the output will have smaller dimension than the input. One can pad the input with zeros, or other values, in order to make the output have the same dimensions as the original input. The output is commonly called feature map (See figure 2.2). The stack of multiple feature maps is the volume output of a convolutional layer.

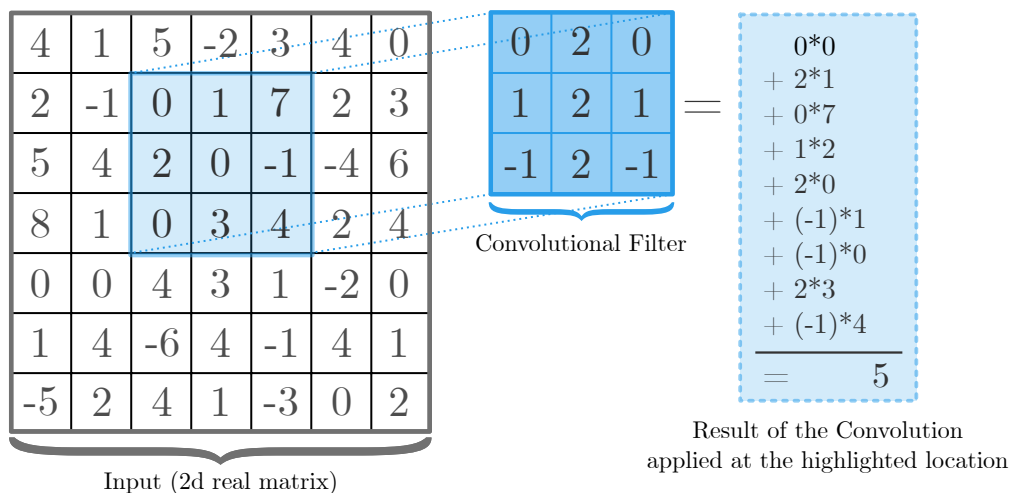


Figure 2.1: Illustration of a Convolutional operation in a 2D matrix.

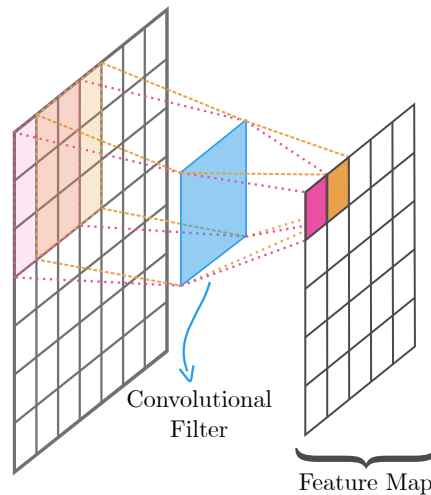


Figure 2.2: Illustration of a feature map. The filter is applied at all valid locations of the input producing the feature map.

As normal Neural Networks layers, convolutional layers are followed by an *activation function*. These activation functions are non-linear functions included between layers, to prevent the model of being one large linear transformation. One of the most used activation function is the Rectified Linear Unit (ReLU), which is defined by:

$$\text{ReLU}(x) = \max(0, x) \quad (2.3)$$

The activation function is applied to all values of the convolutional output.

In conjunction with convolutional layers, the CNNs use *pooling layers* in their architecture. Pooling layers are similar to convolution in the sense that they apply an operation over a window location in an input volume. Instead of a linear product with a filter, as in the convolution, the pooling filter applies an aggregation function over a location. This aggregating operation can be a maximum, minimum, average, etc., over all values in the window. The pooling is used to reduce the spatial dimensions of the input, since it condenses information of neighbor vectors into a single one using some operation. This improves the computational cost of processing the model, and more importantly, it increases the receptive field of follow up layers. This increase in the receptive field size is illustrated in Figure 2.3

During training, the filters in the convolutional layers start to learn patterns from the images. The pooling layers create a hierarchical structure in the network, with filters in starting layers learning simple patterns, such as colors, edges, etc. and filters in subsequent layers learning more complex patterns by aggregating the information of previous layers. A visualization of simple CNN network is presented in Figure 2.4.

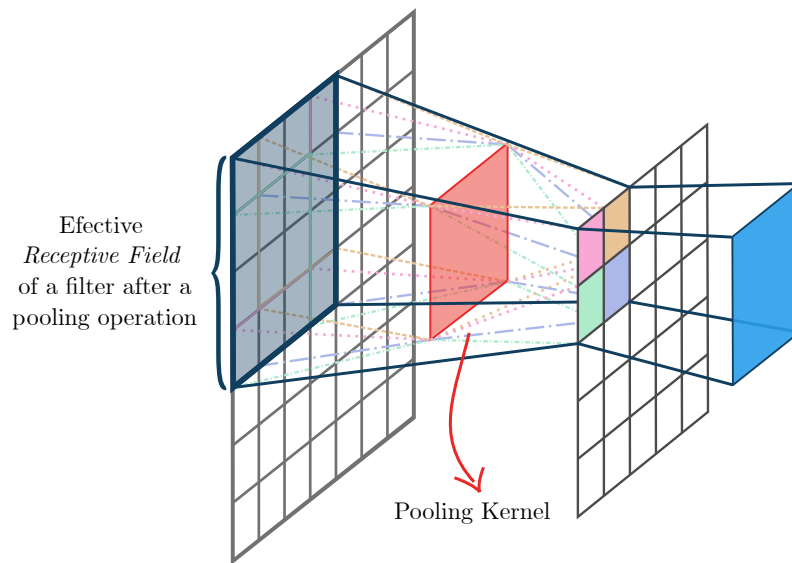


Figure 2.3: The increased *receptive field* of a filter after a pooling operation.

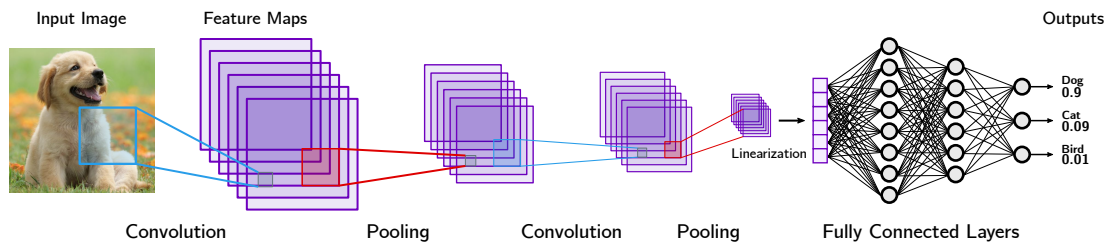


Figure 2.4: Illustration of a simple CNN architecture.

### 2.1.2 Fully Convolutional Networks and UNet

In the CNN models for classification after a sequence of convolutional and pooling layers a linearization of the feature volume is performed and the resulting vector is processed with fully-connected layers, that is, the usual Neural Networks layers (Figure 2.4). This way, the model output a simple vector, that we can interpret as the probabilities of each class.

With the objective of obtaining a more structured output as in the semantic segmentation problem, Long et al. [2015] proposes the Fully Convolutional Network (FCN). The FCN discards the fully-connected layers of the CNN and substitutes with convolutional layers. The fully-connected layers have a fixed size and discard spatial relation over the features when the volumes are linearized, prior to them. This spatial relation is important to the semantic segmentation task in which neighbor pixels are expected to be from the same class. Also, the fixed number of neurons in the fully-connected layers forces the input to have a specific size. Since the CNN architectures utilizes convolutions and pooling, which reduces the height and width of the volume as they are processed by the model, the FCN model needs to upscale the output. This is done by using transposed



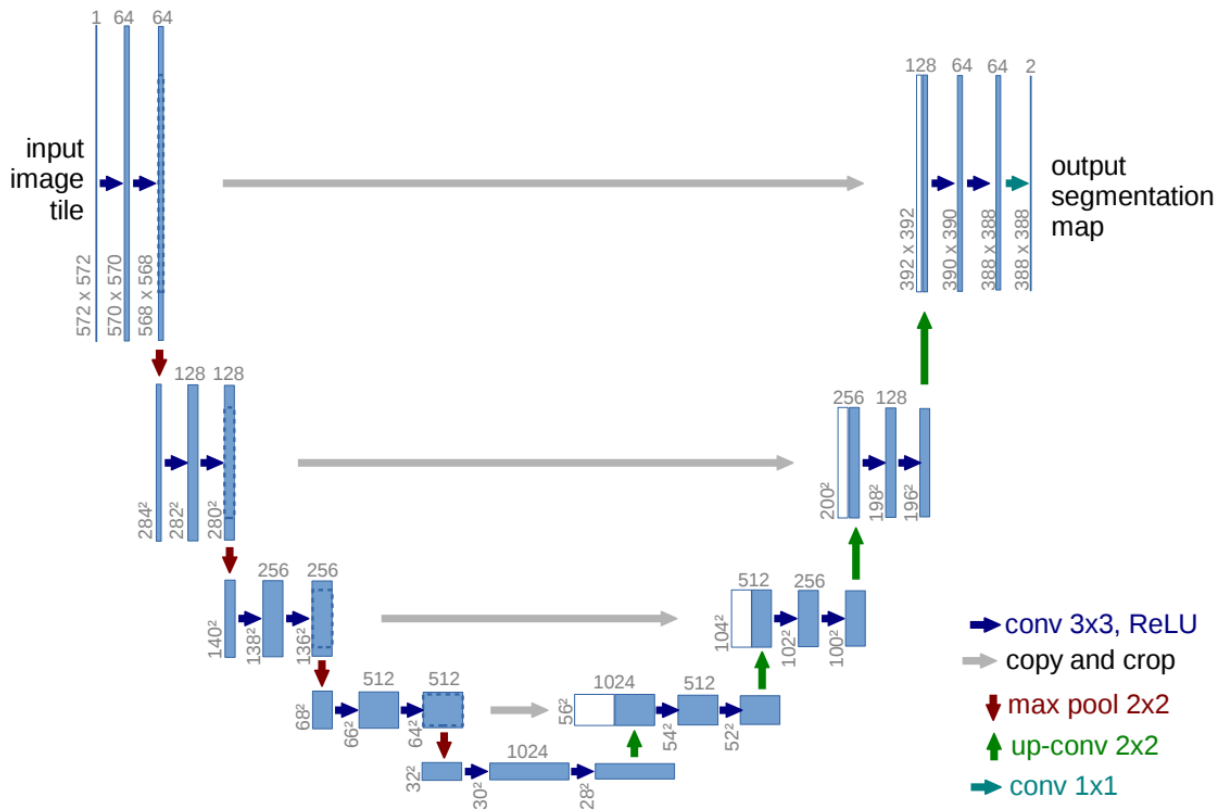


Figure 2.5: Illustration of the original UNet architecture. (Source: [Ronneberger et al. \[2015\]](#))

convolutions that are inserted at the end of the model. Different from a fixed method, such as bilinear interpolation, the transposed convolution can be learned by the model, and this proved to result in better predictions than using those fixed methods. This large upscale usually produce a coarser segmentation due to abrupt increase in dimensions. To minimize this problem, the authors realized a small upscale, then aggregate information of a pooling layer with same dimensions of the resized volume, and afterwards upscale to the final dimensions of the output.

The UNet [[Ronneberger et al., 2015](#)] is a fully convolutional architecture for semantic segmentation that uses an *encoder-decoder* structure (Figure 2.5). That is, the input is processed by the encoder creating a volume with a high feature dimension and usually small spatial dimensions, and then this volume is further *decoded*, gradually expanding its spatial dimension and reducing its feature dimensions, to finally produce the segmentation output. In detail, the UNet is a symmetrical architecture with the same numbers of encoder and decoder blocks, with a singular middle block, and a final convolutional layer with kernel size  $1 \times 1$  that perform pixel classification. An encoder block is composed of two convolutional layers, with ReLU activation, followed by a  $2 \times 2$  max-pooling layer that reduces the spatial dimension by half. The decoder blocks are analogous, they have a transposed convolution, that amplifies the spatial dimensions by two, followed by two

convolutional layers with ReLU activation. One key aspect of this architecture is the use of a type of skip-connection (gray arrows in Figure 2.5). Each decoder has a symmetrical paired encoder and its input is a concatenation of the last decoder block output with the paired encoder output. The authors proposed this architecture initially for segmenting biomedical imaging, and had won two challenges with this type of images in 2015<sup>2</sup> with this network.

## 2.2 Model-Agnostic Meta-Learning (MAML)

The Model-Agnostic Meta-Learning (MAML) [Finn et al., 2017] is an algorithm to train models for fast adaptation. Its agnostic name, is derived from the fact that the algorithm is designed to work with any model or problem (classification, regression, etc.), as long as it uses a gradient based training. The principle is to expose the model to different tasks, such that it can be easily and fastly adapted to a new task. Figure 2.6 illustrates this concept.

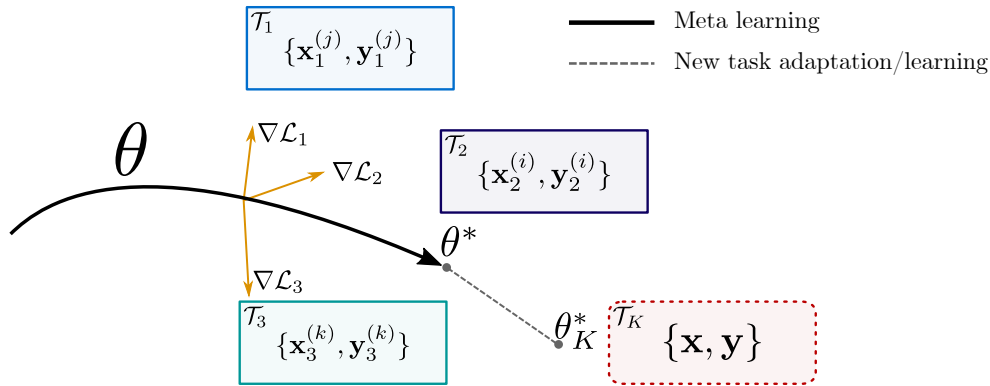


Figure 2.6: MAML Diagram. The parameter  $\theta$  is optimized by training on multiple tasks  $\mathcal{T}_i$ , by following their losses  $\mathcal{L}_i$ , and thus the optimized  $\theta^*$  can be fastly adapted to a new task  $\mathcal{T}_K$ .

The algorithm is generic so it can be applied in multiple scenarios and problems. For simplicity, we will formally define the algorithm for common supervised problems, like classification or segmentation. A task  $\mathcal{T} = \{\mathcal{L}, \mathcal{D}^{sup}, \mathcal{D}^{qry}\}$  is defined by a loss function  $\mathcal{L}$ , and two sets of data points  $\mathcal{D}^{sup}$  and  $\mathcal{D}^{qry}$ . These sets are comprised of pairs in the format  $(\mathbf{x}, \mathbf{y})$ , where  $\mathbf{x}$  is a data sample, and  $\mathbf{y}$  is its label. We assume a distribution over tasks  $p(\mathcal{T})$  that our model  $f$  is desired to be able to adapt to. The algorithm can be summarized by having two nested loops: an outer and inner loop. In the inner loop the

<sup>2</sup><https://lmb.informatik.uni-freiburg.de/people/ronneber/isbi2015/>

model is trained to adapt to a task  $\mathcal{T}_i$  sampled from  $p(\mathcal{T})$ , while in the outer loop, the cumulative error of the tasks is used to train the model.

Formally, we define a model  $f_\theta$  with trainable parameters  $\theta$ . In the inner loop, the parameters used by the model become  $\theta_i$ , which is initialized with the values of  $\theta$ . Then,  $\theta_i$  is updated by one or more steps of gradient descent with training samples from  $\mathcal{D}_{\mathcal{T}_i}^{sup}$  set using,

$$\theta_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_\theta) \quad (2.4)$$

where the step size  $\alpha$  can be a fixed parameter, or meta-optimized as well.

In the outer loop, the model’s parameters  $\theta$  are trained across all tasks  $\mathcal{T}_i$  drawn from  $p(\mathcal{T})$ , by evaluating the performance of the model  $f_{\theta_i}$  on that task by validating on data from the  $\mathcal{D}_{\mathcal{T}_i}^{qry}$  set. Thus the objective in the outer loop can be defined as,

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_\theta)}) \quad (2.5)$$

and the parameter  $\theta$  is optimized with gradient descent following

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i}) \quad (2.6)$$

Note that, by Equations 2.6 and 2.4, when updating the value of  $\theta$  we compute second-order derivatives of the  $\mathcal{L}$  function, which can be computationally costly.

Algorithm 1 summarizes this supervised training version of MAML.

---

**Algorithm 1** Model-Agnostic Meta Learning: Supervised version
 

---

**Require:**  $p(\mathcal{T})$ : distributions over tasks

**Require:**  $\alpha, \beta$ : step size hyperparameters

randomly initialize  $\theta$

**while** not done **do**

  Sample batch of tasks  $\mathcal{T}_i \sim p(\mathcal{T})$

**for all**  $\mathcal{T}_i$  **do**

    Sample batch of datapoints  $S_i = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$  from  $\mathcal{D}_{\mathcal{T}_i}^{sup}$

    Compute  $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_\theta)$  using  $S_i$  and  $\mathcal{L}_{\mathcal{T}_i}$

    Update adapted parameters using gradient descent:  $\theta_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

    Sample batch of datapoints  $Q_i = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$  from  $\mathcal{D}_{\mathcal{T}_i}^{qry}$

**end for**

  Update  $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i})$  using each  $Q_i$  and  $\mathcal{L}_{\mathcal{T}_i}$

**end while**

---

Once the general parameter  $\theta$  is obtained from the meta-training, it can be used as the initial weight of the model for a new task. With this new task, any procedure can be performed to adapt this parameter to the new problem. Most commonly, a simple fine-tuning is performed using the dataset of said task. Given the presumed generality from the  $\theta$  parameter, this fine-tuning is expected to have an easily and rapidly convergence.

## 2.3 Prototypical Networks

Prototypical Networks [Snell et al., 2017], sometimes referred as ProtoNets, is a model proposed for Few-Shot classification. The core idea of this method is to construct a class prototype for each class, and compare this entity to the samples features. The inferred class of a sample is the same class of the entity most similar to its features. Figure 2.7 illustrate the model.

Formally, given a support set  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ ,  $x_i \in \mathbb{R}^D$ ,  $y_i \in \{1, 2, \dots, K\}$  with  $N$  labeled samples, the prototype  $\mathbf{c}_k$  of class  $k$  is defined as:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_{\Phi}(x_i) \quad (2.7)$$

where  $S_k = \{(x_i, y_i) \in S \mid y_i = k\}$  is the subset of examples with class  $k$ , and  $f_{\Phi} : \mathbb{R}^D \rightarrow \mathbb{R}^M$ , is an embedding function with parameters  $\Phi$  (for example, a CNN model). By Equation 2.7, we define the class prototype as the mean vector of the embedded samples of that class.

For inference of the class  $y$  of a query point  $\mathbf{q} \in \mathbb{R}^D$ , the Prototypical Networks create an distribution  $p_{\Phi}$  over classes using a softmax function, defined as follow:

$$p_{\Phi}(y = k | \mathbf{q}) = \frac{\exp(-d(f_{\Phi}(\mathbf{q}), \mathbf{c}_k))}{\sum_i \exp(-d(f_{\Phi}(\mathbf{q}), \mathbf{c}_i))} \quad (2.8)$$

where  $d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}^+$ , is a distance function.

The squared Euclidean Distance  $d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|^2$  and the Cosine similarity  $d(\mathbf{u}, \mathbf{v}) = \mathbf{u} \cdot \mathbf{v} / (\|\mathbf{u}\| \|\mathbf{v}\|)$  are commonly used distance functions.

The loss function for training is the negative log-probability  $J(\Phi)$ :

$$J(\Phi) = -\log p_{\Phi}(y = k | \mathbf{q}) \quad (2.9)$$

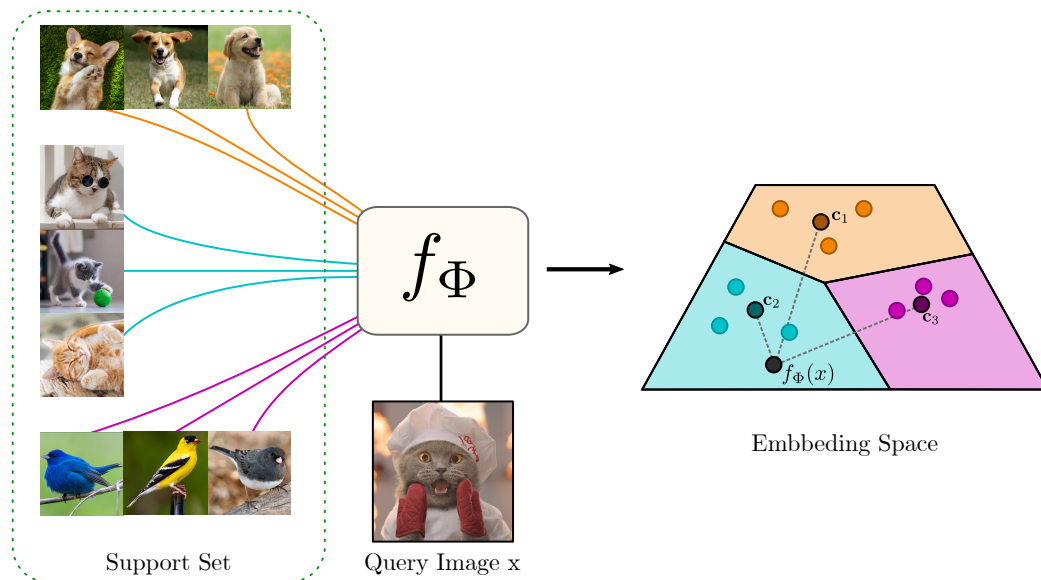


Figure 2.7: Illustration of the Prototypical Networks. The embedding function  $f_\Phi$  embeds the support set, which is used to compute the class prototypes  $c_k$ . The embedded query image is compared to the class prototypes to predict its class.

# Chapter 3

## Literature Review

This chapter will present some works that have approaches similar to the ones in this dissertation, or that were proposed to problems related to the one we are investigating. In section 3.1 we present some works for the problem of Weakly Supervised Semantic Segmentation that uses sparse labels. Afterwards, in section 3.2, we discuss approaches to the problem of Few-shot classification, focused primarily on meta learning methods. Finally, in section 3.3, we present some works to the problem of Few-shot Semantic Segmentation.

### 3.1 Weakly Supervised/Sparse Label Semantic Segmentation

The problem of semantic segmentation with sparse labels is a challenging one. Approaches to this problem can be mostly divided in two main groups. One group that tries to use the sparse labels provided without any form of augmenting it and the second group that tries to reconstruct a dense annotation from the sparse labels.

The works of [Lin et al., 2016; Wang et al., 2018a; Vernaza and Chandraker, 2017] are part of the second group. Lin et al. [2016] is one of the first works to use sparse labels for semantic segmentation. They use a label propagation scheme in conjunction with an FCN for segmentation. This propagation uses the scribble annotation provided and the prediction of the FCN network. They train their model by alternating which part is trained at each iteration. When they fix the propagation model, they train the FCN network, then they fix the parameters of the FCN and train the label propagation scheme using the new predictions of the network. Wang et al. [2018a] uses a principle similar to Lin et al. [2016], but applied to biomedical images. The difference is that the model is trained on full annotations, and then they fine tune for specific images. In this fine tune, the model iteratively refines their segmentation for an image, possibly using scribbles provided by the user interactively. Their training is alternated, as in Lin et al.

[2016], where they fix the network and update the labels, and fix the labels to update the network. Vernaza and Chandraker [2017] also approach the case of semantic segmentation with sparse labeling using a label propagation scheme. The authors propose a two branch pipeline, one of an ordinary segmentation network and the other a label propagation using random-walk. During training, the propagation branch infer a dense label for an image from the sparse annotation provided, while the other branch is a simple segmentation network that tries to segment the image. The inferred labels are then used in a cross-entropy loss to train both branches.

The survey of Tajbakhsh et al. [2020] reviews a collection of deep learning solutions to medical image segmentation problems. This survey includes a section for segmentation with noisy/sparse labels. The methods for sparse labels can be summarized in a selective loss with or without mask completion. A selective loss is a type of loss function that have different weights to unlabeled pixels/voxels, and with this can ignore such pixels when the total cost is computed. All the methods reviewed in the survey use such loss functions, but works like [Zhang et al., 2019; Bai et al., 2018; Cai et al., 2018; Can et al., 2018; Matuszewski and Sintorn, 2018] use some technique to augment the sparse annotations to resemble complete masks. Alternatively, in other works they do not try to reconstruct a complete mask. For example, the works of [Çiçek et al., 2016; Bokhorst et al., 2018] uses a weighted loss, [Silvestri and Antiga, 2018] implies the use of padding in the sparse labels, and [Zhu et al., 2019] uses a quality model to ensure a good segmentation based of the sparse annotation.

## 3.2 Few-shot Classification and Meta-learning

Data can be a scarce and/or expensive resource in some scenarios. In conjunction with a search for more data efficient methods, this motivates the interest of the scientific community in few-shot problems. As one of the simpler problems in computer vision, image classification have been the focus of many methods in literature.

One simple approach to this few-shot problem is to use fine-tuning. That is, we train a model in a large dataset and save the weights. Then, we follow by training the model with these pre-trained (saved) weights on the few labeled samples provided. Some works use domain adaptation, which it can be seen as an extension of this methodology. They use a large dataset called *source domain* and the few-shot dataset that is called *target domain*. The goal of domain adaption is to reduce the shift between the domains. Dong and Xing [2018b] uses a model that is trained for one-shot classification using adversarial domain adaptation.

Another simple approach is to use data augmentation to increase the number of the labeled samples artificially. In the literature we refer to the small set of labeled samples with information of a target class as the *support set*, while the set of images that we want to segment is called *query set*. [Antoniou et al. \[2017\]](#) uses Generative Adversarial Networks (GANs) [[Goodfellow et al., 2014](#)] to data augment the support set of the few shot tasks and then train classifier models in this new set. [Wang et al. \[2018b\]](#) includes a generative model in their framework for few-shot classification. The support set is fed to the generative model that create new samples, which are added to the support set, and then used to train the classifier. If the classifier is differentiable in relation to the loss function, then the whole framework can be trained end-to-end, including the generative model.

Given the limited size of the labeled examples, methods usually tries to extract the maximum of information of this set. In extreme cases, where there are no labeled data (zero-shot), methods resort to external sources, as in the work of [Socher et al. \[2013\]](#). They propose a method for zero-shot image classification where they use a list of words that correspond to possible image classes. They have two embedding methods, one for the words and other for the images, that transfer these data to the same semantic space. In this semantic space they can infer the class of the images based on the log-likelihood of a possible class word.

[Qiao et al. \[2018\]](#) proposes a different approach to the few-shot classification task, where they try to predict the weights of a classifier from the activations of a network. They train a small model that receives activations of a CNN network – the linearized vector of the output of convolutional layers, or the output of penultimate fully-connected layer –, and output the parameters of the classifier layers of the CNN. This model is trained on classes that have a large sample size. For the few-shot classes, they use the few labeled samples to produce the parameters of the classifier layer and thus infer the class of new samples.

Another increasingly used approach to few-shot classification is meta learning methods. The survey of [Hospedales et al. \[2020\]](#) proposes a refined taxonomy to group meta learning methods. However, a simplistic way to group the methods, as seen in [[Lee and Choi, 2018](#); [Yao et al., 2020](#)], suffices in this case. The methods that will be presented can be divided in two groups, the metric learning and optimization approaches. The former group focus on models that learns embedding functions and compute distances in these embedding spaces. The later focus in new methods to direct optimize models in this data regime, and provide good initialization for fine-tuning. The following works can be seen as metric learning approaches: [[Vinyals et al., 2016](#); [Snell et al., 2017](#); [Simon et al., 2020](#)]. On the other hand, the works [[Finn et al., 2017, 2018](#); [Raghu et al., 2019](#); [Nichol et al., 2018](#)] are examples of Optimization/Gradient based approaches. As already explained in Chapter 2, the methods MAML [[Finn et al., 2017](#)] (Section 2.2) and



Prototypical Networks [Snell et al., 2017] (Section 2.3) will be omitted in this chapter.

The metric learning approaches can vary their definitions of embedding space, as well as their transformation functions. Vinyals et al. [2016] proposes the Matching Networks, an attention-based model to guide its predictions. They use embedding functions  $f$  and  $g$  (e.g. CNNs) to encode query and support images, respectively. The probability of a query image class is a weighted mean of the class probabilities of the support set. The weight is calculated by using an LSTM model [Hochreiter and Schmidhuber, 1997] that receives the encoded query and support set. In the work [Simon et al., 2020] the authors presents the Deep Subspace Networks (DSNs), a model that learns to create a  $d$ -dimensional subspace for each class from samples of the support set. They use a CNN to embed all the images from the support set. These embedded images are  $d$ -dimensional vectors that are grouped by the class of their images. Each group is assumed to contain a basis for a subspace, that will become the *class subspace*. After constructing these subspaces, to infer the class of an image, they encode the image with the CNN in a  $d$ -dimensional vector, and compute the distance between the embedded image and its projection in each subspace. This distance is used to compute the class probabilities.

Optimization based meta-learning approaches produce good initial weights to be tuned to the few-shot tasks. By being one of the first methods, the MAML algorithm [Finn et al., 2017] proposed a solid baseline to modifications. Nichol et al. [2018] proposes the Reptile algorithm. It only uses first-order gradient information. Different from MAML, there is a single loop in the meta-training and the global parameter  $\theta$  is optimized in a different manner. A task is sampled and the task specific parameter  $\theta_i$  is obtained by gradient updates using the task loss and examples. Then  $\theta$  is simply updated by using  $\theta \leftarrow \theta + \epsilon(\theta_i - \theta)$ , where  $\epsilon$  is the step size. Finn et al. [2018] presents a probabilistic version of MAML. The  $\theta$  parameters are not the weights of a network, but instead they define a distribution over such weights. During training, in the inner loop they sample weights to be updated using information from the test set of the task. The outer loop loss, in addition to the usual loss, also includes a term of Kullback-Leibler Divergence between the distribution updated with information from the train set, and a distribution with information from the test set. The work of Raghu et al. [2019] is a modification of the MAML algorithm. The novelty proposed by the authors is that in the *inner* loop of MAML, instead of updating all the  $\theta_i$  parameters, they only update the parameters related to the network head, that is, the components of the network that are correspondent to a classifier, in this case. With these change, the network converge to parameters that are reusable to multiple tasks, and reduce the computational cost of training.

### 3.3 Few-Shot Semantic Segmentation

Similar to the few-shot classification problem, to exploit information from the support set have an important role in the semantic segmentation case. Multiple works try to insert this information directly into the model’s processing flow.

Many works use a two branch structure, where one branch is responsible to extract information from the support samples and this info is fused into the other branch that processes the query images. [Shaban et al. \[2017\]](#) implements a simple network to produce weights based on the support set, that are used as parameters of a logistic regression prediction. A pair of image/label from the support set is used to guide the segmentation of a query image. The support pair is fed to a CNN to produce weights  $w, b$ . The features extracted from the query image using an FCN model generate the segmentation by being processed by a convolution layer that uses the weights  $w, b$ . [Dong and Xing \[2018a\]](#) uses a two branch model, where one network produces prototypes of each class, as the ProtoNets. The first branch uses the support set and query image to produce prototypes, which are used for image classification in this branch. The second branch encodes the query image and fuses it with the prototypes of the other branch. From this fusion probabilities maps are created, and then are further used to generate the final predictions. In [Hu et al. \[2019\]](#), the authors introduce an attention-based model. Their two branch model is highly interconnected. Each branch extracts features from different sources: one from the query images and the other from masked support images. It uses attention modules at multiple layers of the model, which receive both the query and support features. At last, a recurrent network is used to access information of all samples in the support set and produce the final prediction. [Zhang et al. \[2020\]](#) is another two branch model. One branch, called *guidance* is used to extract feature vectors from both query and support images. From the support features, they use the support label to construct class representative vectors using a masked average pooling. These vectors are further used to compute a similarity map - using cosine distance - from the query features. Finally, the similarity map is fused with the query features from the second branch, that produces the final predictions.

There are other approaches that use a single network to face the few-shot semantic segmentation problem. [Wang et al. \[2019\]](#) proposes a direct adaptation of the Prototypical Network [[Snell et al., 2017](#)]. They use a CNN to produce feature vectors of the images in the support set and compute the prototypes for each class using masked average pooling. They use the cosine distance as their distance function. One novelty introduced by the authors is the inclusion, during training, of an *alignment* loss, where the prototypes are computed from the query image, and the support set is the segmentation target. [Rakelly et al. \[2018\]](#) proposed the Guided Networks (or, Guided Nets), the first algorithm for few-shot sparse segmentation. Although, it fuses information from the support set in

Table 3.1: Presented Literature Works

Work	Semantic Segmentation	Few-Shot	Sparse Annotations
Lin et al. [2016]	✓		✓
Vernaza and Chandraker [2017]	✓		✓
Zhang et al. [2019]	✓		✓
Bai et al. [2018]	✓		✓
Cai et al. [2018]	✓		✓
Can et al. [2018]	✓		✓
Matuszewski and Sintorn [2018]	✓		✓
Çiçek et al. [2016]	✓		✓
Bokhorst et al. [2018]	✓		✓
Silvestri and Antiga [2018]	✓		✓
Zhu et al. [2019]	✓		✓
Dong and Xing [2018b]		✓	
Antoniou et al. [2017]		✓	
Wang et al. [2018b]		✓	
Socher et al. [2013]		✓	
Qiao et al. [2018]		✓	
Vinyals et al. [2016]		✓	
Snell et al. [2017]		✓	
Simon et al. [2020]		✓	
Finn et al. [2017]	?	✓	?
Finn et al. [2018]	?	✓	?
Raghu et al. [2019]	?	✓	?
Nichol et al. [2018]	?	✓	?
Dong and Xing [2018a]	✓	✓	
Hu et al. [2019]	✓	✓	
Zhang et al. [2020]	✓	✓	
Wang et al. [2019]	✓	✓	
Rakelly et al. [2018]	✓	✓	✓
WeaSeL Method (Ours)	✓	✓	✓
Prototype Method (Ours)	✓	✓	✓

query features, this model uses a single feature extraction network. The Guided Nets use a pre-trained CNN backbone to extract features of both the support set and the query image. The support features are averaged through a masked pooling using the sparse annotations provided for the images, and further globally averaged across all the support images available. The single averaged support feature is then tiled to produce a spatial compatible tensor with the dimensions of the query features. The tiled support feature is fused with the query features with a simple point-wise multiplication operation. The fused features are further processed by a small convolutional segmentation head that gives the final predictions.

Table 3.1 summarizes the works presented in this Chapter and how our proposed methods fit in the literature. In this table, works marked with a *question mark* symbol (?) were not defined originally and/or tested in this scenarios but can be potentially used.

# Chapter 4

## Methodology

In this chapter we define the problem of few-shot semantic segmentation with sparse labels, and present two approaches to solve this problem. The problem definition is in Section 4.1, our first proposed method, based in gradient descent, is in Section 4.2, and the second approach is presented in Section 4.3. These approaches are derived from the MAML and Prototypical Networks, which were presented in Chapter 2.

### 4.1 Problem Definition

First, let us define a segmentation task  $\mathcal{S}$ . A segmentation task is a tuple  $\mathcal{S} = \{\mathcal{D}^{sup}, \mathcal{D}^{qry}, T\}$ , where  $\mathcal{D}$  is a dataset with partitions  $\mathcal{D}^{sup}$ , called support set, and  $\mathcal{D}^{qry}$ , named query set, such that  $\mathcal{D}^{sup} \cap \mathcal{D}^{qry} = \emptyset$ . A dataset  $\mathcal{D}$  is a set of pairs  $(\mathbf{x}, \mathbf{y})$ , where  $\mathbf{x} \in \mathbb{R}^{H \times W \times B}$  is an image with dimensions  $H \times W$  and  $B$  bands/channels, and  $\mathbf{y} \in \mathbb{N}^{H \times W}$  is the semantic label of the pixels in the image.  $T$  is the task target class considered as foreground for the segmentation. Note that with this definition, we only consider binary tasks, where we have one target class  $T$  and the remainder are considered background. Since multiclass problems, with  $C$  classes, can be easily converted in  $C$  binary problems, we only investigate the binary cases. For simplicity of notation, we sometimes define a segmentation task as a pair  $\mathcal{S} = \{\mathcal{D}, T\}$ , with implicit support/query partitions of  $\mathcal{D}$ .

A few-shot semantic segmentation task  $\mathcal{F}$  is a specific type of segmentation task. It is also a tuple  $\mathcal{F} = \{\mathcal{D}^{sup}, \mathcal{D}^{qry}, T\}$ , but the samples of  $\mathcal{D}^{sup}$  have their labels sparsely annotated, and the labels in  $\mathcal{D}^{qry}$  are absent or unknown. Moreover, the number of samples  $k = |\mathcal{D}^{sup}|$  is a small number (e.g. less than 20), thus we also call a few-shot task, a  $k$ -shot task.

Now our problem of few-shot semantic segmentation with sparse labels can be defined as follow. Given a set of segmentation tasks  $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n\}$ , and a few-shot task  $\mathcal{F}$ , we want to segment the images from the  $\mathcal{D}_{\mathcal{F}}^{qry}$  using information from tasks  $\mathcal{S}_i$ , and information from the  $\mathcal{D}_{\mathcal{F}}^{sup}$ . It holds that no pair of image/semantic label of  $\mathcal{F}$  is present

in any task  $\mathcal{S}_i$  in either *qry* or *sup* partition. This implies that, there is no semantic information of the target objects of the  $\mathcal{F}$  task, other than the sparse annotations of  $\mathcal{D}_{\mathcal{F}}^{sup}$  samples.

## 4.2 Gradient-based Semantic Segmentation from Sparse Labels

We adapt the supervised MAML algorithm, presented in Section 2.2, to our problem of few-shot semantic segmentation with sparse labels. We call this approach **Weakly-supervised Segmentation Learning** (WeaSeL), and an illustration can be seen in Figure 4.1.

We will use a different definition for a task  $\mathcal{T}$ . Now, a meta-task  $\mathcal{T}$ , is a *segmentation task*, as defined in Section 4.1, with a loss function commonly used in segmentation problems: the Cross-Entropy loss, which can be defined for a single pixel  $j$  as,

$$\mathcal{L}_{CrossEntropy}(j) = - \sum_i^C y_i^j \log f_{\theta}(x)_i^j \quad (4.1)$$

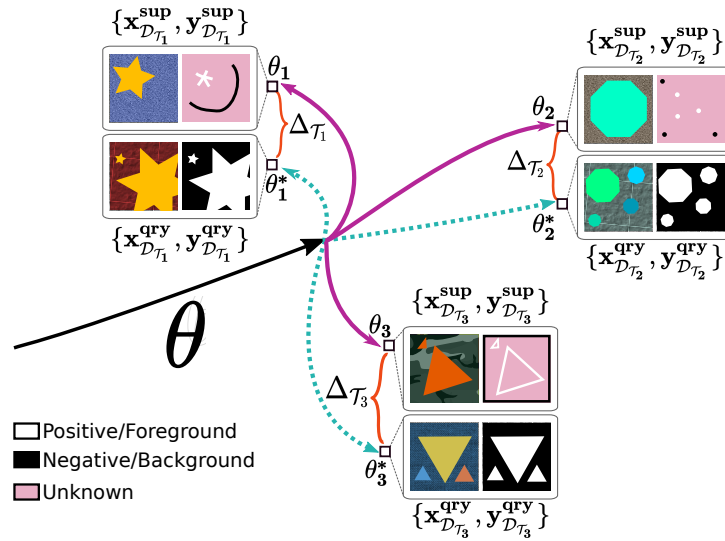
where  $C$  is the number of classes,  $y_i^j$  is the true probability of a class  $i$  for the pixel  $j$ , and  $f_{\theta}(x)_i^j$  is the predicted probability by the model  $f_{\theta}$  for the class  $i$  to the  $j$  pixel. Since we compute this cost to a single pixel, to apply to a whole image the loss in Equation 4.1 is averaged over all pixels, that is for an image  $x$ , the Cross Entropy is defined as,

$$\mathcal{L}_{CrossEntropy}(x) = - \frac{1}{N} \sum_{j \in x} \sum_i^C y_i^j \log f_{\theta}(x)_i^j \quad (4.2)$$

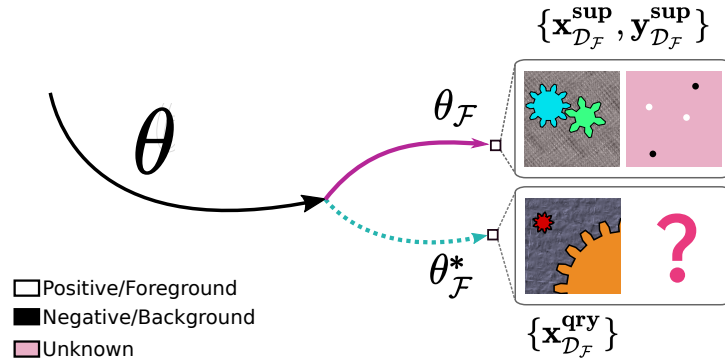
where  $N$  is the total number of pixels in  $x$ .

Since our few-shot problem uses sparse labels, we simulate this types of annotations in the meta-tasks. That is, for all  $\mathcal{T}_i \sim p(\mathcal{T})$ , the labels of samples in  $\mathcal{D}^{sup}$  are randomly converted to a sparse version of themselves (see Section 5.2 for details) . This is applied so the model can be trained in conditions similar to the one in which it will be fine-tuned. With this, we expect that it will learn to predict dense labels from sparse annotations, and more easily adapt to the few-shot task.

We use the Algorithm 1 for the meta-training, but with the distribution  $p(\mathcal{T})$  over our new segmentation meta-tasks. With this, in the inner loop the loss is computed using the simulated sparse annotations of the support set of a task, and the outer loss using the dense labels of the query set of a task  $\mathcal{T}_i$ .



(a) Visualization of the meta-training process. The parameters  $\theta_i$  are optimized using sparse labels from support sets. The optimal  $\theta_i^*$  would be obtained if dense labels were presented in meta-training, as the ones in the query set used to compute the task outer loss. We expect that the model learns to intrinsically minimize this difference between parameters ( $\Delta_{\mathcal{T}}$  in the figure), and thus fastly adapt to the few-shot task.



(b) Illustration of the fine tuning step. The meta-optimized  $\theta$  is supervised trained with the sparse annotated samples of the few-shot support set. The labels of the query are unknown, i.e., not seen by the model.

Figure 4.1: Illustration of the WeaSeL method with toy examples.

Given that the labels are sparse in the inner loop of meta-training, we modify the Cross-Entropy loss to work in this scenarios as follow,

$$\mathcal{L}_{SelectiveCE} = -\frac{1}{N} \sum_j \sum_i^C w^j y_i^j \log f_{\theta}(x)_i^j \quad (4.3)$$

where  $j$  is a pixel,  $N$  is the total number of valid/labeled pixels, and  $w^j$  is an indicator, where  $w^j = 0$ , if  $j$  has an unknown label and  $w^j = 1$ , otherwise. That is, we ignore pixels with unknown labels by using this weighted loss, and average the loss for all valid pixels.

After the meta-training we have to adapt the model to the few-shot task. In order to accomplish this, we perform a simple fine-tuning with the samples from support set of the few-shot task  $\mathcal{F}$ . That is, we use the few labeled pairs  $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\mathcal{F}}^{sup}$  to train the model for a number of epochs in a supervised manner using the  $\mathcal{L}_{SelectiveCE}$  loss.

### 4.3 Prototypical Seeds for Sparse Segmentation

The proposed method for semantic segmentation based on the Prototypical Networks is a straightforward adaptation of the original method described in Section 2.3. It uses the same premise of constructing a prototype vector to each class, with the distinction that prototypes are computed using the labeled pixels instead of whole image instances.

Given a support set  $S = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ , where  $x_i \in \mathbb{R}^{H \times W \times B}$  is an image with height  $H$ , width  $W$  and  $B$  channels, and  $y_i \in \mathbb{R}^{H \times W}$  is a label image with the semantic class of each pixel in  $x_i$ . Since  $y_i$  can be sparse, the possible values of a pixel  $j$  in  $y_i$  are in the set  $\{0, 1, 2, \dots, K\}$ , where  $K$  is the total of classes and 0 represents the unknown class.

Similar to equation 2.7, in this adaptation we define the vector  $\mathbf{c}_k$ , of a class  $k$  as

$$\mathbf{c}_k = \frac{1}{N_k} \sum_{(x_i, y_i) \in S} \sum_j [f_\Phi(x_i) \odot \mathbb{1}_k(y_i)]^j \quad (4.4)$$

where  $f_\Phi$  is our embedding function parametrized with  $\Phi$  (a CNN),  $\odot$  is point-wise multiplication, and  $\mathbb{1}_k(y_i) \in \{0, 1\}^{H \times W}$  is a mask matrix where each value is defined as

$$\mathbb{1}_k^j(y_i) = \begin{cases} 1, & \text{if } y_i^j = k \\ 0, & \text{otherwise} \end{cases}$$

And  $N_k = \sum_{y_i} \sum_j \mathbb{1}_k^j(y_i)$  is the total number of pixels of the class  $k$ , across all the support set  $S$ . In contrast with classical Prototypical Networks, which map an entire image to a single vector in embedding space, our embedding function  $f_\Phi$  maps each pixel of an image to a corresponding vector in embedding space. We then compute our prototype vector  $\mathbf{c}_k$  as the mean vector of the features of all pixels of a class existent in the support set. (See Figure 4.2).

The inference is the same of the original Prototypical Networks, but applied to a pixel in the image. Formally, the probability  $y^j$  of a pixel  $j$  in a query image  $\mathbf{q} \in \mathbb{R}^{H \times W \times B}$  belonging to a class  $k$  is computed as follow:

$$p_\Phi(y^j = k | \mathbf{q}) = \frac{\exp(-d(f_\Phi(\mathbf{q})^j, \mathbf{c}_k))}{\sum_i \exp(-d(f_\Phi(\mathbf{q})^j, \mathbf{c}_i))} \quad (4.5)$$

where  $d$  is the squared euclidean distance:  $d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|^2$ .

Similar to the WeaSeL method case, given the presence of unknown labeled pixels in training we modify our loss function to ignore such pixels. This way we define our new loss function  $J(\Phi)$  as follows,

$$J(\Phi) = -\frac{1}{|Q|} \sum_{(\mathbf{x}, \mathbf{y}) \in Q} \sum_{j \in \mathbf{x}} \sum_{k=1}^K \log p_\Phi(\mathbf{y}^j = k | \mathbf{x}^j) \quad (4.6)$$

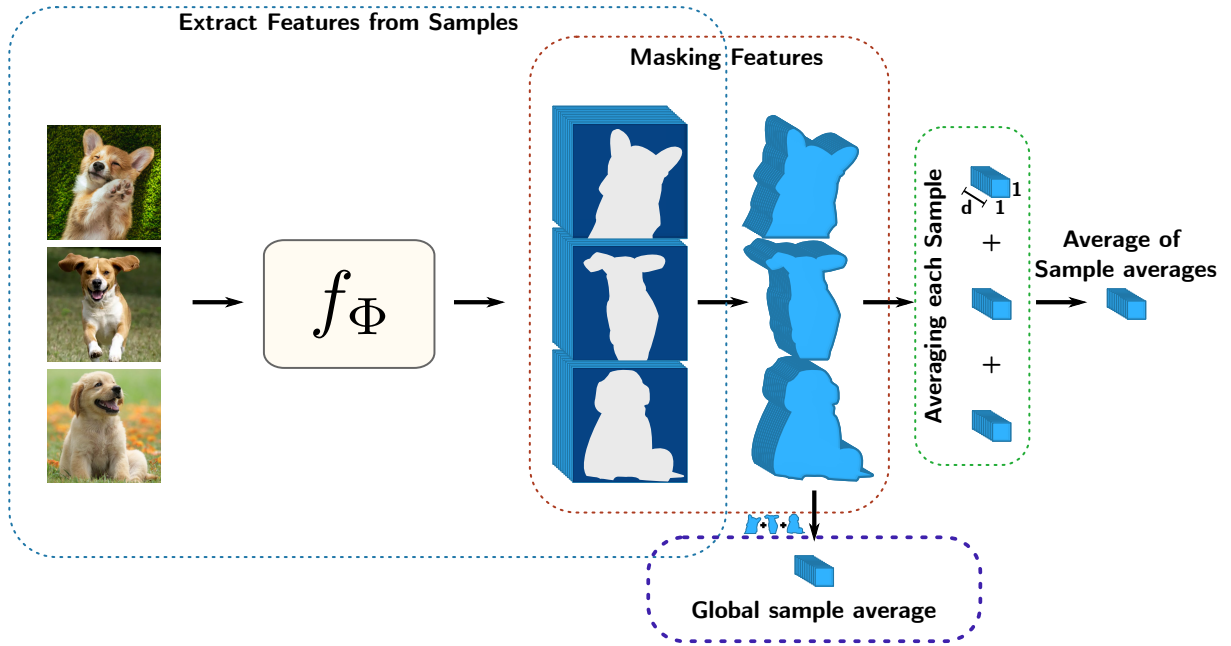


Figure 4.2: Illustration of a Masked Average Pooling. Our process differ after masking the features. A common Masked Average Pooling, will average these features to each sample, and then average these averages (right). Ours will create the *global sample average* (bottom) by considering all pixels.

where  $Q$  is the set images used to compute the loss,  $j$  is a pixel coordinate and  $k$  represents a class. Note that  $k$  starts from 1, thus not considering the unknown class  $k = 0$ . We use  $p_\Phi$  as defined in Equation 4.5.

Given equations 4.4 and 4.5, the model  $f_\Phi$  is trained using an episodic training strategy. This strategy resemble the training algorithm of WeaSeL, and is presented in Algorithm 2. It uses the same distribution over tasks  $p(\mathcal{T})$  as the first method, and uses the generated sparse annotations of the meta-tasks in training. At each iteration a batch of tasks is sampled, and for each task  $\mathcal{T}_i$  a support set  $S_i$  is constructed and used for training.



---

**Algorithm 2** Prototypical Networks: Few-shot segmentation training
 

---

**Require:**  $p(\mathcal{T})$ : distributions over tasks

randomly initialize  $\Phi$

**while** not done **do**

  Sample batch of tasks  $\mathcal{T}_i \sim p(\mathcal{T})$

**for all**  $\mathcal{T}_i$  **do**

    Sample a support set  $S_i = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$  from  $\mathcal{D}_{\mathcal{T}_i}^{sup}$

    Convert  $y_j \in S_i$  to sparse annotations

    Compute  $\mathbf{c}_k$  using  $S_i$ , for all  $k$  using equation 4.4

    Sample query batch  $Q_i = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_b, \mathbf{y}_b)\}$  from  $\mathcal{D}_{\mathcal{T}_i}^{qry}$

    Compute the loss  $J(\Phi)$  as defined in equation 4.6, using  $Q_i$ .

    Update  $\Phi$  using gradient descent and  $\nabla J$

**end for**

**end while**

---

# Chapter 5

## Experimental Setup

In this chapter, we present the configurations used for our experiments. In Section 5.1 we succinctly present the datasets used. In Section 5.2 the evaluated sparse labels annotation styles are listed. Next, in Section 5.3, we introduce the CNN architecture used, and in Section 5.4 the baselines, protocol, and metrics are presented.

All the code for the experiments were written in the Python3 language. For the models, we use the Pytorch<sup>1</sup> framework and the Torchmeta<sup>2</sup> module. In relation to the machine, all the experiments were performed on Ubuntu SO, 64-bit Intel i9 7920X machine with 64GB of RAM memory and a GeForce RTX 2080 TI/Titan XP GPU (only one of the GPUs were used during an experiment).

### 5.1 Datasets

We choose to test our methods in areas with real applications. As mentioned, medical imaging and remote sensing have several applications to semantic segmentation. Moreover, these two areas have some similarities that differ from them from others. First, the images used in their tasks are very distinct from daily images taken with a cellphone camera, for example. This already hinders the knowledge transfer from other generic domains or the use of pre-trained models on large datasets, such as ImageNet. Another common aspect of medical imaging and remote sensing areas is the limitation of availability of these types of images by different factors. Medical images require several agreements due to patient privacy, as well as the necessity of a specialist to provide precise annotations. Remote sensing images have a high monetary cost to acquire since satellites and/or drones are commonly used to collect them. In addition, they usually require a specialist to label the collected data.

These aspects led us to choose these two areas for the experiments. Also, this

---

<sup>1</sup><https://pytorch.org>

<sup>2</sup><https://github.com/tristandeleu/pytorch-meta>

choice provided a large variety of tasks to validate our methods.

### 5.1.1 Medical Datasets

We use a total of eight medical datasets in our experiments. From these datasets, four are Chest X-Rays datasets (CXR), two are Mammographic X-Rays (MXR), and two are Dental X-Rays (DXR), and will be briefly presented below.

#### 1.A. JSRT Database [Shiraishi et al., 2000]

The JSRT Database [Shiraishi et al., 2000] is a collection of 247 chest radiographs initially proposed for lung nodules identification. Labeled masks for lung, clavicle, and heart structures were obtained from van Ginneken et al. [2006]. A task is created for each anatomical structure.

All the images have a resolution of  $2048 \times 2048$  and 12 bit pixel resolution, and are gray scale.

Examples are show in Figure 5.1(a).

#### 1.B. Montgomery Dataset [Jaeger et al., 2014]

The Montgomery Dataset [Jaeger et al., 2014] is a set of chests X-rays collected from patients in Montgomery County, Maryland, USA. There are 138 frontal X-rays, from which 58 are from cases of Tuberculosis - the initial use case of the dataset was to classify the presence of this disease. Alongside with information of the patient, for each X-ray, there are binary masks for segmentation of each lung.

All the X-rays are 12 bit gray scale images, and either have size  $4020 \times 4892$  or  $4892 \times 4020$ . Examples are shown in Figure 5.1(b).

#### 1.C. Shenzhen Dataset [Jaeger et al., 2014]

The Shenzhen Dataset [Jaeger et al., 2014] were publicized along with the Montgomery dataset. This set is comprised of chests X-rays collected from patients in Shenzhen, China. There are a total of 662 frontal X-rays, from which 336 are from cases of Tuberculosis. There is also binary masks for segmentation of lungs.

The size of the X-rays vary, but average a  $3000 \times 3000$  resolution, and are gray scale. Examples are shown in Figure 5.1(c).

#### 1.D. NIH-labeled [Tang et al., 2019]

This is a subset of the original NIH-labeled dataset [Wang et al., 2017]. The original dataset is comprised of 108,948 frontalview X-ray images of 32,717 unique patients, and labeled with NLP for 14 different diseases.

The subset used in this experiments will be named simply NIH-labeled or XLSor-NIH. This dataset proposed in Tang et al. [2019] is comprised of 100 chest X-rays from the original NIH with manually annotated lung masks for these X-rays.

All images have a spatial resolution of  $512 \times 512$  and are gray scale. Examples can be seen in Figure 5.1(d).

### 1.E. OpenIST Chest X-Rays

The OpenIST Chest X-Rays dataset<sup>3</sup> is a set of X-rays collected from the following original domain: <http://www.chestx-ray.com/index.php/education/normal-cxr-module-1>. These images were used to train medical students in recognizing a normal X-ray. There are in total 225 chest X-ray images, with binary masks for the lungs.

The images are gray scale with 8-bit resolution and their sizes are not fixed. Examples in Figure 5.1(e).

### 1.F. LIDC-IDRI-DRR dataset [Oliveira et al., 2020]

Is a dataset derived from LIDC [Armato III et al., 2011]. This dataset is composed of flattened 2D Digitaly Reconstructed Radiographs (DRR) computed from chest CT-scans, with automatically generated ribs annotations [Oliveira et al., 2020].

All the 835 images in the dataset are gray scale, and have size  $512 \times 512$ . Examples of the scans and labels are show in Figure 5.1(f).

### 1.G. MIAS database [Suckling et al., 1994]

The Mammographic Image Analysis Society (MIAS) is a research group in the UK with interest in mammograms. Their dataset is composed of 322 digitized mammograms, grayscaled and with a resolution of  $1024 \times 1024$ . The original dataset only provides labels of location and size of nodules in the images, but we had access to label masks that segment the pectoral muscles and the breast in each image.

Example of samples are shown in Figure 5.2(a).

### 1.H. INbreast database [Moreira et al., 2012]

The INbreast database is a collection of 410 images collected from woman in the Breast Center located in the Centro Hospitalar de S. Joao [CHSJ], Porto, Portugal. From these 410 images, only 200 are from Mediolateral Oblique (MLO) view (a side view of the breasts), and have labeled pectoral muscles for them. We also obtained labels for the breasts.

---

<sup>3</sup><https://github.com/pi-null-mezon/OpenIST>

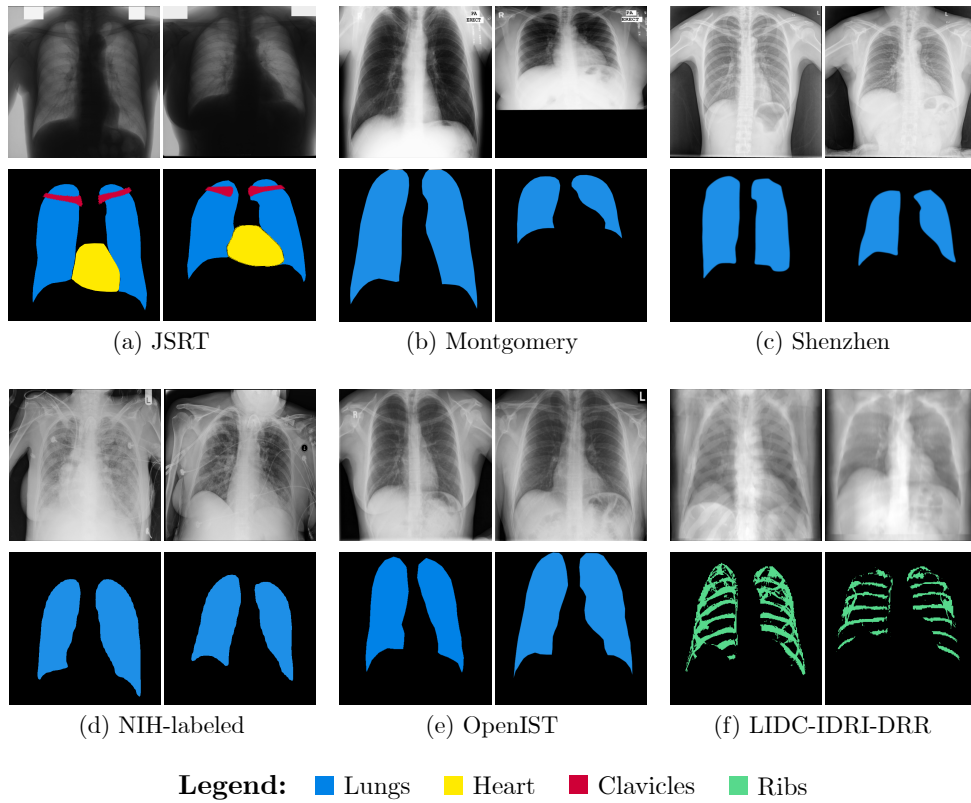


Figure 5.1: Examples from the Chest X-Ray Datasets.

All images are gray level with 14 bit resolution and their sizes are either  $3328 \times 4084$  or  $2560 \times 3328$ , depending on the patient. Examples can be seen in Figure 5.2(b).

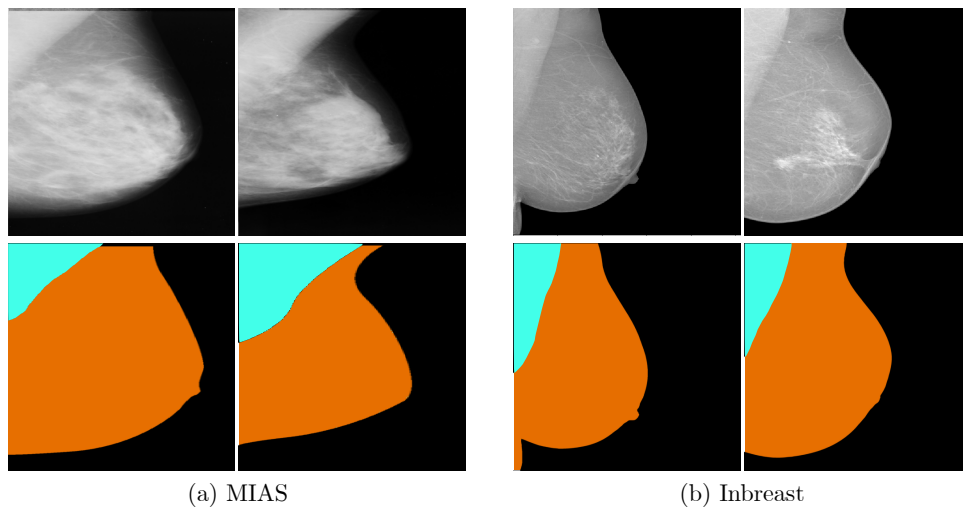


Figure 5.2: Examples from the Mammography Datasets.

### 1.1. Panoramic Dental X-rays [Abdi et al., 2015]

The Panoramic Dental X-rays dataset is a set of panoramic dental X-rays of 116 patients, taken at Noor Medical Imaging Center, Qom, Iran. The images were

manually segment by three specialists from which label masks for the mandibles were generated.

All the images are gray scale and have a size of approximately  $2900 \times 1250$  pixels. Examples of sample images and labels are presented in Figure 5.3(a).

### 1.J. UFBA-UESC Dental Images Dataset [Silva et al., 2018]

The UFBA-UESC Dental Images Dataset (or, simply, UFBA Dataset) is composed of a series of panoramic X-ray dental images. There is a total of 1500 images with annotated teeth labels, with a variety of cases of dental problems and/or formations defects.

All the images are gray scale with dimensions of  $2440 \times 1292$  pixels. Examples are shown in Figure 5.3(b).

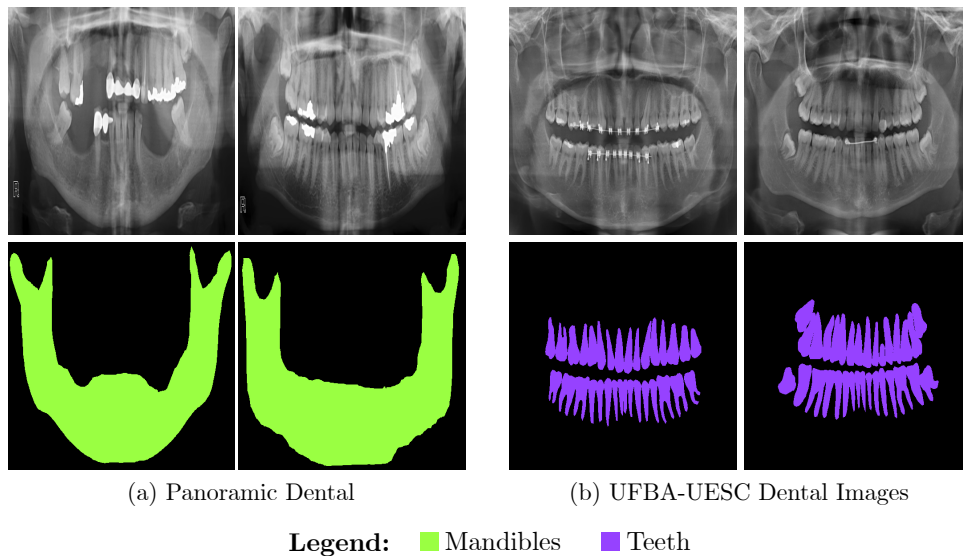


Figure 5.3: Examples from the Dental X-Rays Datasets.

## 5.1.2 Remote Sensing Datasets

The Remote Sensing datasets can be divided in two groups: Rural (or Agriculture) and Urban datasets. There are two rural datasets, and two urban datasets, that will be divided in three tasks groups. The datasets are listed below:

### 2.A. Brazilian Coffee

Is a dataset comprised of 4 large satellite images from 4 municipalities of the state of Minas Gerais, Brazil - one satellite image for each county. The four municipalities

are: Arceburgo, Guaranésia, Guaxupé and Montesanto. In the satellite images only three bands were considered, namely the Red, Green, and Near Infrared bands. Along these images, a binary ground truth label image is provided, with positive value representing a coffee crop and negative value representing the background.

For evaluation purpose, each of the images were cropped in non-overlapping patches of size  $256 \times 256$ , and only crops with a percentage of 25% or more of pixels of coffee were maintained for training and validation of the models. Given that each county has distinct geographical features, which lead to coffee plantation distinctions, we consider each municipality a different task.

Examples of patches of this dataset and their respective ground truths can be seen in Figure 5.4(a)-(d).

## 2.B. Orange Orchards

This dataset is comprised of satellite images from an Orange Orchard located at the municipality of Ubirajara, São Paulo, Brazil. In the satellite images four bands are presented, Red, Green, Blue, and Near InfraRed, but only three of the bands were used: all but the Blue band. In addition to the images, annotation masks of the orange plantations were provided, making two classes, namely, Oranges and Background.

Each of the satellite images were cropped in non-overlapping patches of size  $384 \times 384$ , and only crops with a percentage of 10% or more of pixels of plantation were maintained for training and validation of the models. With the large crop size we used a small percentage of the pixels to have a similar minimum number of positive pixels to the coffee tasks. Since this dataset is focused in a single region and have only one interest class, we use it as a single task in our experiments.

Examples of patches of this dataset and their respective ground truths can be seen in Figure 5.4(e).

## 2.C. ISPRS Potsdam<sup>4</sup> and Vaihingen<sup>5</sup>

The ISPRS Potsdam and Vaihingen datasets were proposed for a semantic segmentation competition in the remote sensing scenario. Each one consists of a collection patches of varying size from a true orthophoto mosaic (38 and 33 patches, for Potsdam and Vaihingen, respectively). It is a series of satellite photo of the cities of Potsdam and Vaihingen, in Germany. Only a part of the patches have an annotated ground truth - 24 and 16 for Potsdam and Vaihingen, respectively -, as the remainder stays secret for the organizers, so they can evaluate the competitors entries.

---

<sup>4</sup><https://www2.isprs.org/commissions/comm2/wg4/benchmark/2d-sem-label-potsdam/>

<sup>5</sup><http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-vaihingen.html>



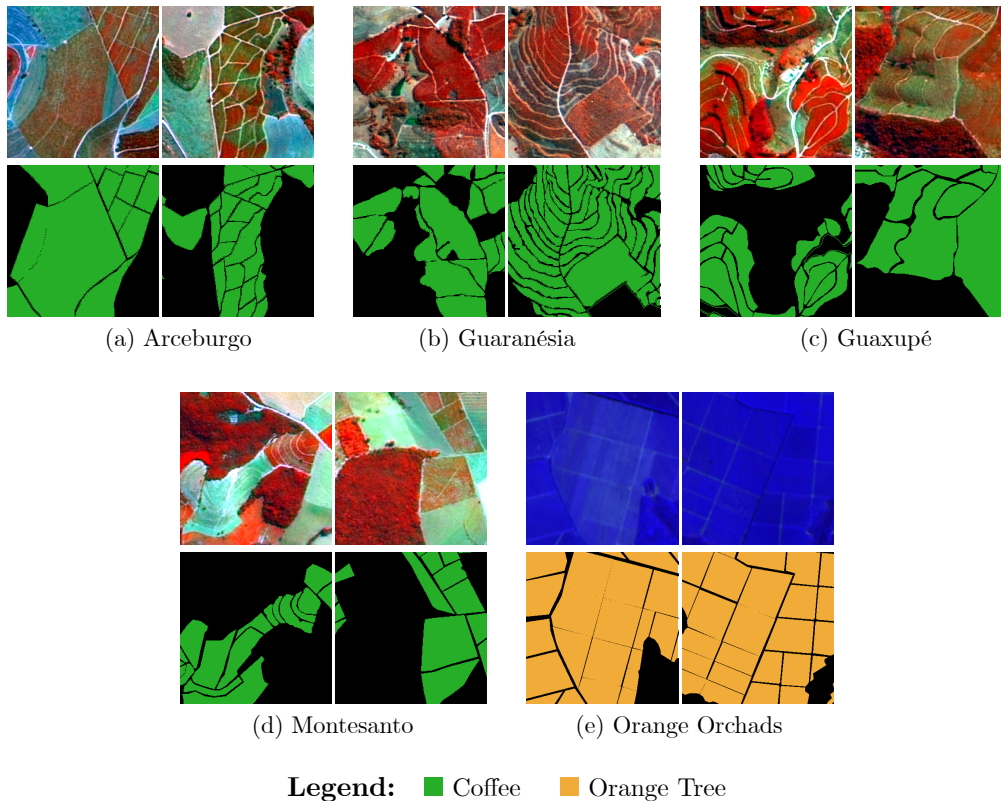


Figure 5.4: Examples from the Brazilian Coffee and Orange Orchards Dataset.

There are different types of patches in the dataset. For Potsdam, there are 3 true orthophotos with different channels information: IRRG images (InfraRed-Red-Green channels), RGB (Red-Green-Blue channels), and RGBIR (Red-Green-Blue-Infrared channels). For Vaihingen, there is only one type of true orthophoto that is a NIRG (Near infrared-Red-Green channels) image. For both cities, there is also a Digital Surface Model (DSM). Only one type of orthophoto were used for each city, for Vaihingen the one available and for Potsdam the RGB image.

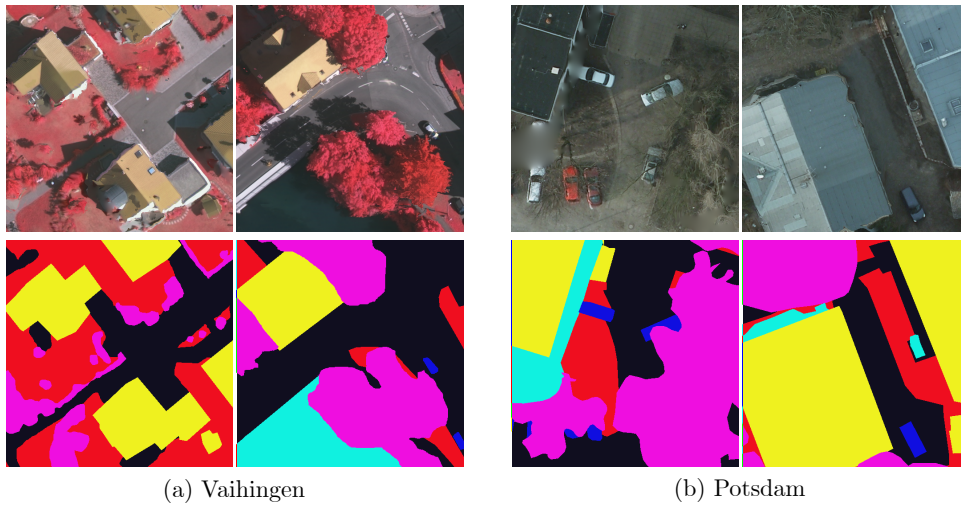
The images in these datasets are also large. The patches of Potsdam all have the same dimension of  $6000 \times 6000$ , while the patches in Vaihingen vary with the smaller dimension being no less than 1200 pixels. For evaluation, all patches were simply cropped from the selected true orthophoto in smaller non-overlapping patches of size  $448 \times 448$ , with no further processing.

Since the datasets contains 6 classes (Impervious surfaces, Building, Low vegetation, Tree, Car, and Clutter/background), we binarize the patches to create different tasks. For a task, we select a class to be positive, while all the remaining are considered negative/background. If a patch has less than a percentage of pixels of positive class, they are discarded for training/validation of the task with that target class. For the the class Car which is comprised of small objects that occupy a small area in the image, any percentage of positive pixels were sufficient to not discard the patch. Meanwhile, for the remainder classes, only patches with more than 25%



of positive pixels were not discarded.

Example of patches and their original ground truth labels are presented in Figure 5.5



**Legend:**    ■ Building    ■ Low vegetation (Grass)    ■ Tree  
              ■ Impervious surfaces (Road)    ■ Car    ■ Clutter/background

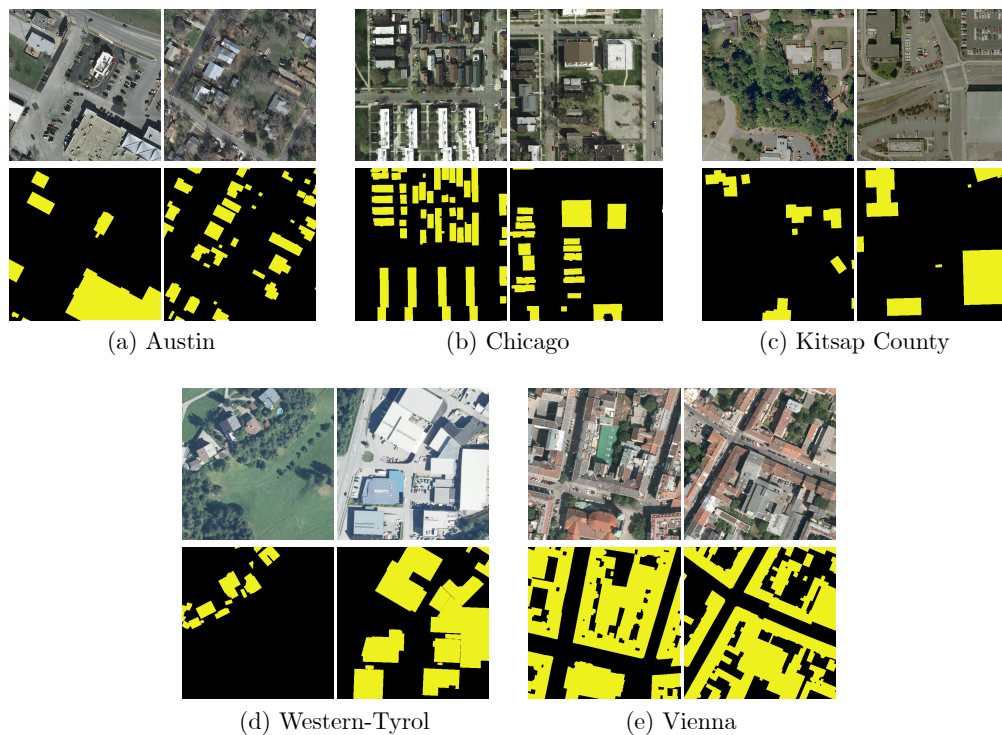
Figure 5.5: Examples from the ISPRS Datasets.

## 2.D. INRIA dataset [Maggiori et al., 2017]

The INRIA dataset [Maggiori et al., 2017] is an urban aerial image dataset of cities in different parts of the world. This dataset was used in a competition and only a subset of the cities are labeled, while others are held out for the internal evaluation. There are 5 labeled cities/regions, which are: Austin (TX, USA), Chicago (IL, USA), Kitsap County (WA, USA), Western-Tyrol (Austria), and Vienna (Italy). For each city, there are 36 RGB tiles of size  $5000 \times 5000$  with a respective binary labeled mask with positive values representing the *building* class and negative the *non-building* class.

Similar to the other remote sensing datasets, for evaluation, each tile were cropped in non-overlapping patches of size  $500 \times 500$ , and patches with less than a percentage of positive pixels — 25% of the image for Kitsap County patches and 40% for the other cities — were discarded. We consider each of the five cities as a different task in our experiments.

Example of patches of the different regions are presented in Figure 5.6



**Legend:** ■ Building ■ Non-Building

Figure 5.6: Examples from the INRIA Dataset.

## 5.2 Types of sparse annotation

In the experiments we evaluate five types of sparse annotation, namely: *points*, *grid*, *contours*, *skeletons*, and *regions*. As mentioned, we simulate these annotations from the original dense labels of an image. Visual examples of these annotations are show in Figure 5.7.

We can describe each type of annotation and explain how the sparse annotations are generated as follow:

**I. Points Annotation:** This type of labeling simulate an annotator alternately picking pixels from the foreground and background classes.

To generate this type of annotation, we use a parameter  $n$  and randomly choose  $n$  foreground and  $n$  from background pixels, while the remainder is set as unknown.

**II. Grid Annotation:** In this labeling process, the annotator receives a pre-selected collection of pixels of the image. All these pixels are initially assumed to be from the background class. Then the annotator select the pixels which it considers to be from the foreground. The pre-selected pixels are disposed in a grid pattern.

This type of labeling is generated by using a parameter  $s$ . Then a random pixel  $p_0$  is selected within the following rectangular region: {upper left corner:  $(0, 0)$  and bottom right corner:  $(s, s)$  }. Afterwards, a grid is created from this  $p_0$  position with  $s$  spacing horizontally and vertically. Pixels outside the grid are set as unknown.

**III. Contours Annotation:** This labeling style is the process where the annotator denotes the inner and outer boundaries of foreground objects. It is most useful for

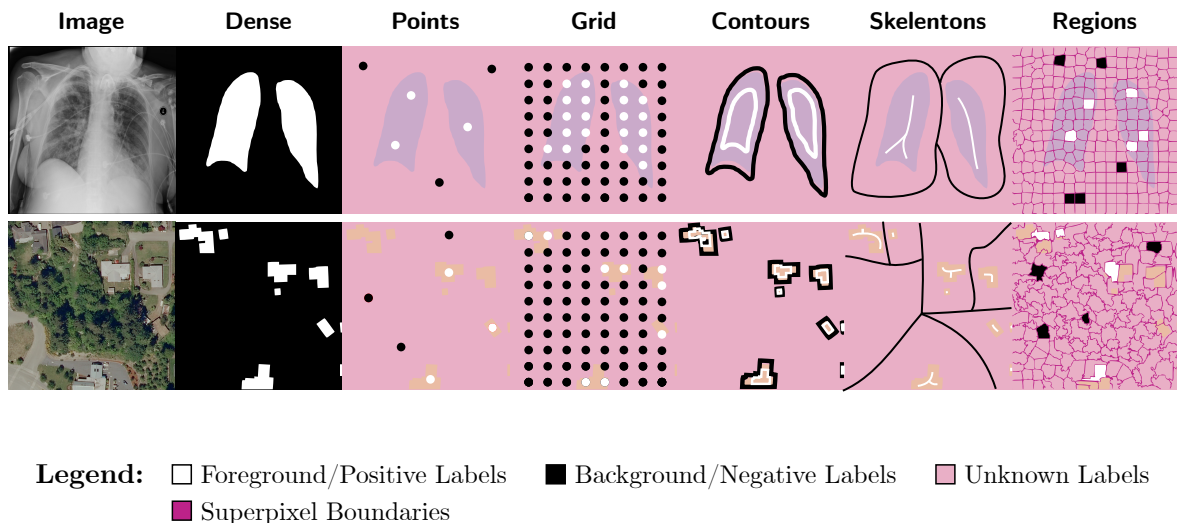


Figure 5.7: Illustration of the types of sparse annotations used. Annotations are illustrative and upscaped for better visualization.

cases where a single connected foreground object is present. If more than one object is present, the outer boundaries can create needlessly complex annotations.

We generate these annotations by using morphological operations on the original binary dense labels. We use an erosion operation followed by a marching squares algorithm<sup>6</sup> to find the inner contours. To the outer contours, we use a dilation operation on the original label mask, and the same marching squares algorithm. Additionally, we use a parameter  $d$  that determines the density of the sparse annotation. After the contours are obtained, we maintain only the segments of the contours that respect the  $d$  threshold. A final dilation on the contours is applied to add thickness to the annotations.

**IV. Skeleton Annotation:** This style of labeling resembles an annotator drawing a scribble roughly at the center of the foreground objects that more or less approximates the object form. The same process is applied to the background.

This annotations are generated using the *skeletonize algorithm*<sup>7</sup> in the binary dense label masks. This returns the skeletons of the foreground objects. The same process is applied to the negative dense label masks to obtain the skeleton of the background class. An additional dilation is applied to add thickness to the skeletons. We use a parameter  $d$  to control the density of the annotation. We generate random binary blobs (using this<sup>8</sup> function) that occupy a  $d$  percentage of the image space and use them to mask the computed skeletons.

**V. Regions Annotations:** This type of annotation represents the process of an annotator appointing classes to *pure superpixels*. We define a *pure superpixel* as a connected set of pixels of the same class. The annotator is provided with the superpixels of the image and appoints the class of a subset of pure superpixels.

To generate these annotations, first we compute the superpixels of the images using the SLIC algorithm [Achanta et al., 2012] with empirically chosen parameters for each dataset. From the computed pure superpixels set, we randomly selected a  $d$  percentage of the superpixels from foreground and a  $d$  percentage of superpixels from the background class.

---

<sup>6</sup>[https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.find\\_contours](https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.find_contours)

<sup>7</sup><https://scikit-image.org/docs/stable/api/skimage.morphology.html#skimage.morphology.skeletonize>

<sup>8</sup>[https://scikit-image.org/docs/stable/api/skimage.data.html#skimage.data.binary\\_blobs](https://scikit-image.org/docs/stable/api/skimage.data.html#skimage.data.binary_blobs)

### 5.3 miniUNet architecture

The network model used in all experiments – Baselines, WeaSeL, and Prototypical – is a simplified version of the UNet architecture [Ronneberger et al., 2015]. We will call it miniUNet since it is a smaller version of the original network. Hardware limitations implied in the choice of smaller network architectures. Since the choice of the network is not imposed for our method, we opt to use this modification of the UNet architecture, which is a staple model in the literature used in several applications. Also, it usually easily converges without a hefty hyperparameter tuning, allowing an easy and fast experiment setup.

The miniUNet network is comprised of three encoder blocks, a center block, three decoder blocks and an  $1 \times 1$  convolution layer that functions as a pixel-classification layer. The network blocks configuration can be seen in Table 5.1 – in the table,  $C$  is the number of input channels an image have (e.g.  $C = 3$  for RGB images), the Input/Output Features is the number of feature dimensions that the input/output volume have (e.g. the *Encoder Block 1* receives an image of size  $h \times w \times C$  and outputs a volume of size  $\frac{h}{2} \times \frac{w}{2} \times 32$ ). Similar to the UNet architecture, skip connections are present in this model. This means that each decoder block receives as input the concatenation of the last block output and the corresponding encoder output. For example, the *Decoder Block 1* receives as input the concatenation of the output volume of *Decoder Block 2* and the output volume of *Encoder Block 1*.

Table 5.1: Descriptions of the miniUNet Blocks

Block Name	Encoder Blocks (1, 2, 3)	Center Block
Layers	Conv $3 \times 3$	Dropout
	Bath Norm.	Conv $3 \times 3$
	ReLU	Bath Normalization
	Conv $3 \times 3$	ReLU
	Bath Norm.	Conv $3 \times 3$
	ReLU	Bath Normalization
	MaxPool $2 \times 2$	ReLU
		Transposed Conv $2 \times 2$
<b>Input Feat./ Output Feat.</b>	$C/32, 32/64, 64/128$	128/128
Block Name	Decoder Blocks (3, 2)	Decoder Block (1)
Layers	Dropout	Dropout
	Conv $3 \times 3$	Conv $3 \times 3$
	Bath Normalization	Bath Normalization
	ReLU	ReLU
	Conv $3 \times 3$	Conv $3 \times 3$
	Bath Normalization	Bath Normalization
	ReLU	ReLU
	Transposed Conv $2 \times 2$	
<b>Input Feat./ Output Feat.</b>	256/64, 128/32	32/32

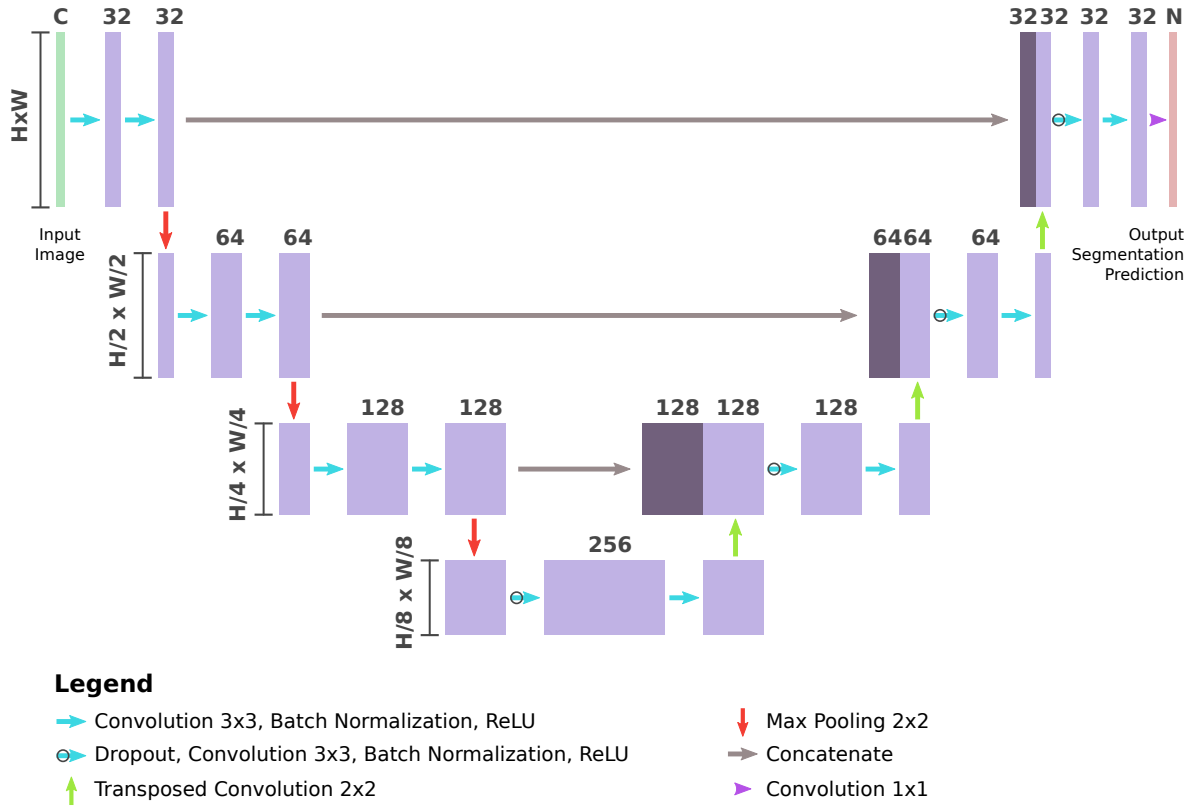


Figure 5.8: Illustration of the miniUNet architecture. The upper numbers represents the feature dimension of the volumes, while on the side is the spatial dimensions.

Unlike the original architecture, we pad the images with zeros prior to the convolutions so the spatial dimensions after the operation is the same as the input. This way, only the poolings and transposed convolutions affect the spatial dimensions of the volume during a forward pass in the network. A visualization of this architecture is provided in Figure 5.8.

When used in Prototypical method, since we want to generate  $d$ -dimensional feature vectors, the last layer of the network is ignored, and the output is gathered from the last decoder block. That is, the embedding function  $f_{\Phi}$  for the images is the miniUNet model excluding the last layer (the  $1 \times 1$  Convolutional layer). Consequently, the embedding space of our prototypes have 32 dimensions.

## 5.4 Evaluation Protocol

### 5.4.1 Baselines

We use two baselines for comparison with our approaches: 1) *From Scratch* and 2) *Fine-Tuning*. Given our few-shot semantic segmentation problem parameters in the form of the set of segmentation task  $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_n\}$ , and a few-shot task  $\mathcal{F}$ , we define our baselines as follow:

- **From Scratch:** Given our miniUNet network, we perform a simple supervised training with the Few-shot task support set ( $\mathcal{D}_{\mathcal{F}}^{sup}$ ). We use the same Cross-Entropy loss ignoring unlabeled pixels as our cost function (Equation 4.3). The Adam optimizer [Kingma and Ba, 2017] was used, with the same parameter used in the training of our methods.
- **Fine-tuning:** We use the miniUNet architecture. We selected one task  $\mathcal{S}_i$  from our tasks set, and perform a supervised training with the  $\mathcal{D}_{\mathcal{S}_i}^{sup}$ . Once finished the training on  $\mathcal{S}_i$ , we perform the fine-tune (a supervised training) using the  $\mathcal{D}_{\mathcal{F}}^{sup}$  set. Again, the same Cross-Entropy loss function (Equation 4.3) is used with the same parametrized Adam optimizer used for our methods.

### 5.4.2 Protocol and Metrics

In order to assess the performance of our methods in a certain setting, we employ a Leave-One-Task-Out methodology. That is, all but the pair (*dataset, class*) chosen as the Few-shot task ( $\mathcal{F}$ ) are used in the Meta-Dataset – the set of segmentation tasks  $\mathcal{S}$  –, reserving  $\mathcal{F}$  for the tuning/testing phase. This strategy serves to simultaneously hide the target task from the meta-training, while also allowing the experiments to evaluate the proposed algorithm and baselines in a myriad of scenarios. Moreover we will divide our tasks in four groups to perform the experiments. These groups are:

- **Medical Tasks:** All the medical datasets (Section 5.1.1) and their classes are used for these tasks, totaling 14 tasks.



- **Agriculture Tasks:** Only the rural datasets (Section 5.1.2[2A-2B]) are used for these tasks. There are 5 tasks in total (4 from the Brazilian Coffee and 1 from the Orange Orchards dataset).
- **ISPRS Tasks:** The ISPRS datasets (Section 5.1.2[2C]) were used. For the tasks, as explained, binary versions with each class as target were used (excluding the Clutter class), thus totaling 10 tasks.
- **INRIA Tasks:** The INRIA dataset (Section 5.1.2[2D]) was used for these tasks, totaling 5 tasks, one for each city in the dataset.

For each method we used different number of epochs, empirically chosen, in each of their training phases. In Table 5.2 we show these numbers. They differ mostly due to training time. The Remote Sensing datasets are, in general, larger than the Medical datasets, and this made the training process (that include validation) more time-consuming.

Table 5.2: Number of epochs for training the methods in different experiments. *#Epochs Tuning* is the number of epochs trained with the Few-Shot Task samples.

Method	Medical Experiments		Remote Sensing Experiments	
	#Epochs Pre/Meta-Training	#Epochs Tuning	#Epochs Pre/Meta-Training	#Epochs Tuning
WeaSeL	2000	80	200	40
Prototypical	2000	-	200	-
Fine-Tuning	200	80	100	80
From Scratch	-	80	-	100

In respect to other hyperparameters, for all experiments, we use the Adam optimizer [Kingma and Ba, 2017] with learning rate 0.001, weight decay 0.0005, and momentum 0.9. Our batch size was set to 5. The number of tasks sampled for the inner loop of WeaSeL and Prototypical was set to 6 in Medical and 4 in Remote Sensing experiments. This was done, in general, due to memory constraints and, in the Remote Sensing experiments, the total number of tasks in each set of experiments.

We use a 5-fold cross validation protocol in the experiments. Each dataset had a training and validation partition for each fold, with 80% of the images used for training and 20% for validation. Once fix the experiment fold, the support sets for the tasks are obtained from the training partition of the dataset, while the query sets are the entire validation partition.

All images and labels are resized to  $256 \times 256$  for remote sensing images, and  $128 \times 128$  for medical images, prior to being fed to the models. This was due to our infrastructure limitations and done in order to standardize the input size and minimize the computational cost of the methods, especially the memory footprint of WeaSeL method, since it computes second derivatives on high-dimensional outputs.

The metric within a fold is computed for all images in the query set according to the dense labels, and is averaged in relation to the images in that fold. The final values reported in Chapter 6 are computed by averaging the performance across all folds.



As for the metric used, we selected the Jaccard score, also known as Intersection over Union (IoU), of the validation images. It is a common metric for semantic segmentation and can measure the similarity between the ground truth and the segmentation of the model. The IoU can measure the similarity between any two sets. Thus, given two sets of points  $A$  and  $B$ , the Jaccard score  $J(A, B)$  is computed as follow:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (5.1)$$

# Chapter 6

## Results and Discussion

In this chapter we present the results of our experiments and a brief discussion about them. We grouped the experiments by research questions that motivate it. First, in Section 6.1, we compare the results of the methods and baselines using multiple sparse annotations and their versions using dense labels. In Section 6.2 we compare different sparse annotations in relation to the number of inputs that an user gives and the score obtained. Finally, in Section 6.3 we compare the number of epochs of fine tuning the few-shot task and the score of our methods.

### 6.1 Few-shot Semantic Segmentation: Sparse vs Dense labels

In this section we present the results of our methods in multiple few-shot tasks in the Medical and Remote Sensing scenarios. We vary the number of shots, and for each type of sparse annotation we vary their parameters as well.

With these experiments we answer our first research question about the impact of using sparse annotations in scenarios few-shot. Also, since we use multiple types of sparse annotations in the experiments, we can visualize how the different styles perform with the evaluated methods.

We will only present some of the results in this section, omitting some redundant results similar to some task included in this section. As a rule for *this section*, we present the results grouped in graphs by sparse annotation type and number of shots (i.e., the number of labeled images in  $\mathcal{D}_{\mathcal{F}}^{sup}$ ). Dashed lines in the graphs are the scores of the methods trained with dense annotations. Additional results can be seen in the Appendix A.

## 6.1.1 Medical Tasks

### 6.1.1.1 Chest X-Ray Datasets

For the evaluated CXR Datasets Tasks (5.1.1 [1.A - 1.E]) we observed a tendency in the results. First, one that holds for, practically, all methods and scenarios, is that with more data better scores are obtained. That is, with large support sets (higher  $k$ -shots) and more sparse labeled pixels the models predicted more precise segmentation.

A second observed result is that the WeaSeL method appears to be the best approach to tasks with a significant discrepancy to other tasks in their category. We observed this in the *JSRT Lungs* (Figure 6.1) and the *JSRT Heart* (Figure 6.2). The *JSRT* dataset is visually the most different of the CXR datasets (as seen in Figure 5.1) moreover, the *Heart* class is only present in this dataset, so this task does not have a direct parallel in other datasets. This discrepancy in both the semantic space and domain from other tasks can explain the inferior results of the *Heart* task in relation to the *JSRT Lungs* task.

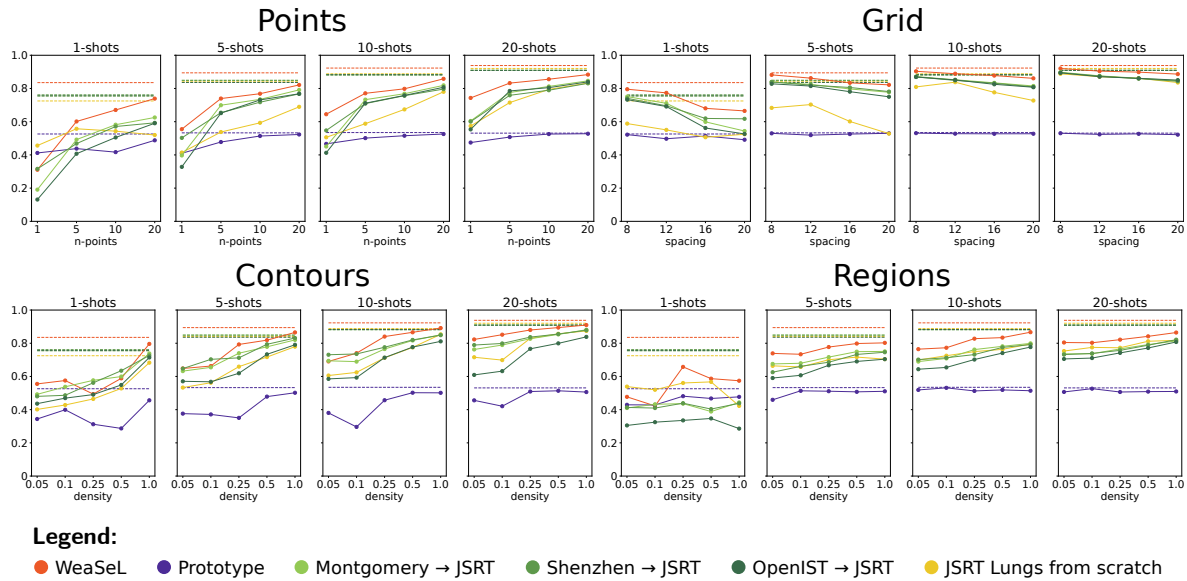


Figure 6.1: Jaccard Score of experiments with JSRT Lungs Task.

For the remainder tasks, we observe that either the Prototypical method or some Fine-tuning baseline is the best performer. Since some datasets are similar visually, fine-tuning for a task from a model trained in a similar dataset is a known viable solution that works well in these cases. These results are visible for the *OpenIST*, *Montgomery*, and *NIH-Labeled Lungs* tasks (Figures 6.3, 6.4, and 6.5, respectively). Fine-tuning from similar tasks (OpenIST, Shenzhen, or Montgomery) yields the best jaccard scores in most

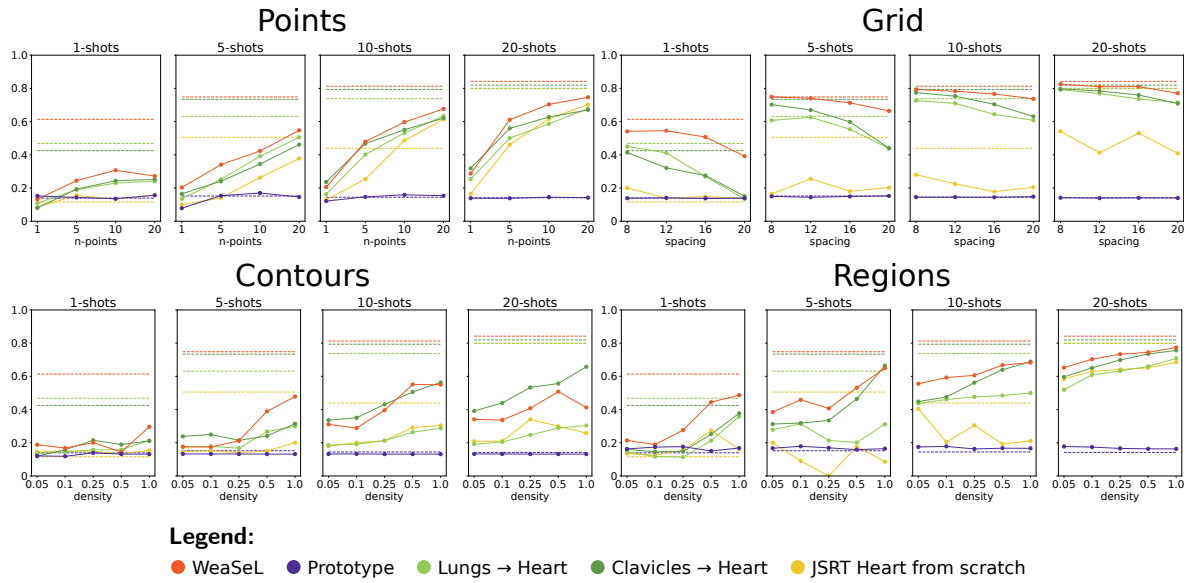


Figure 6.2: Jaccard Score of experiments with JSRT Heart Task.

cases. Also, we observe that the Prototypical is consistently comparable to these fine-tuned baselines.

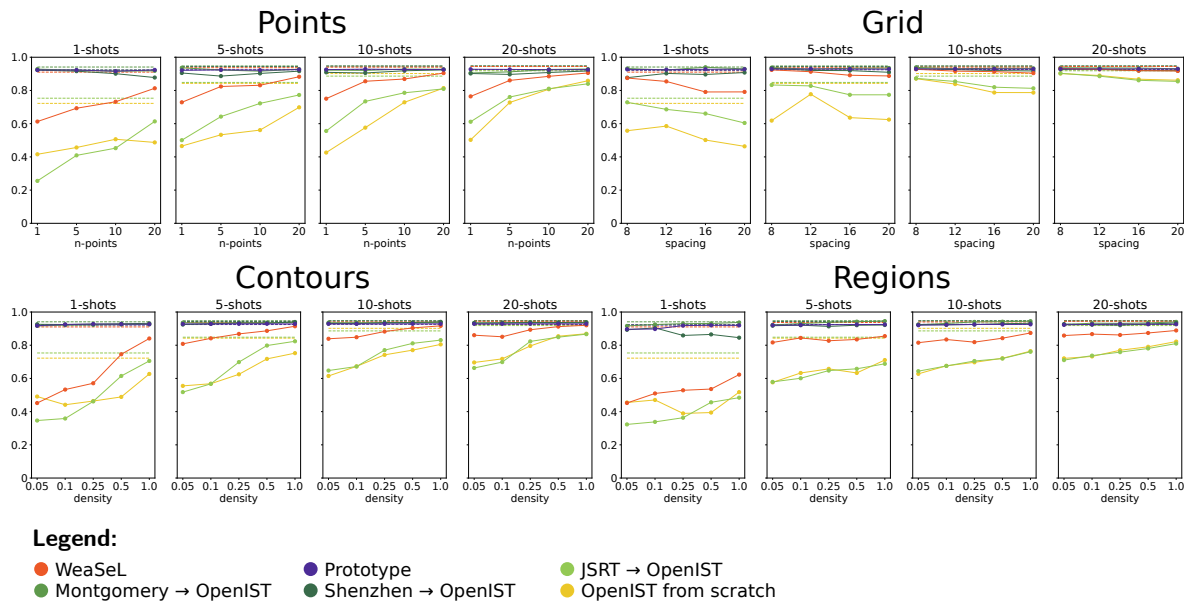


Figure 6.3: Jaccard Score of experiments with OpenIST Lungs Task.

We observe that the methods, as expected, tend to approach their score obtained when trained with dense labels as the sparsity of the annotation decreases. But, even in yet extremely sparse scenarios, such as 20-points annotation that represent less than 1% of the total pixels in the image, the gap between the methods trained on sparse annotations and their dense label versions is small, as can be seen in most of the experiments. The WeaSeL method often achieves the same results with dense annotations when using sparse annotations for tuning. However, the only method that practically is indifferent to the type or sparsity of the annotations is the Prototypical method. Since the method takes the

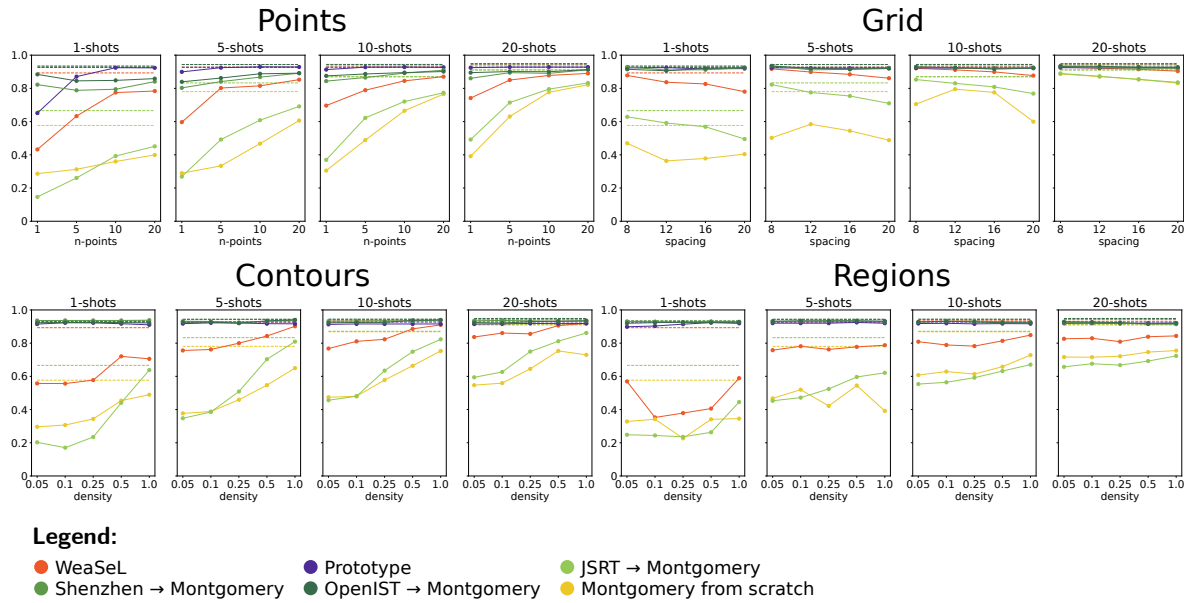


Figure 6.4: Jaccard Score of experiments with Montgomery Lungs Task.

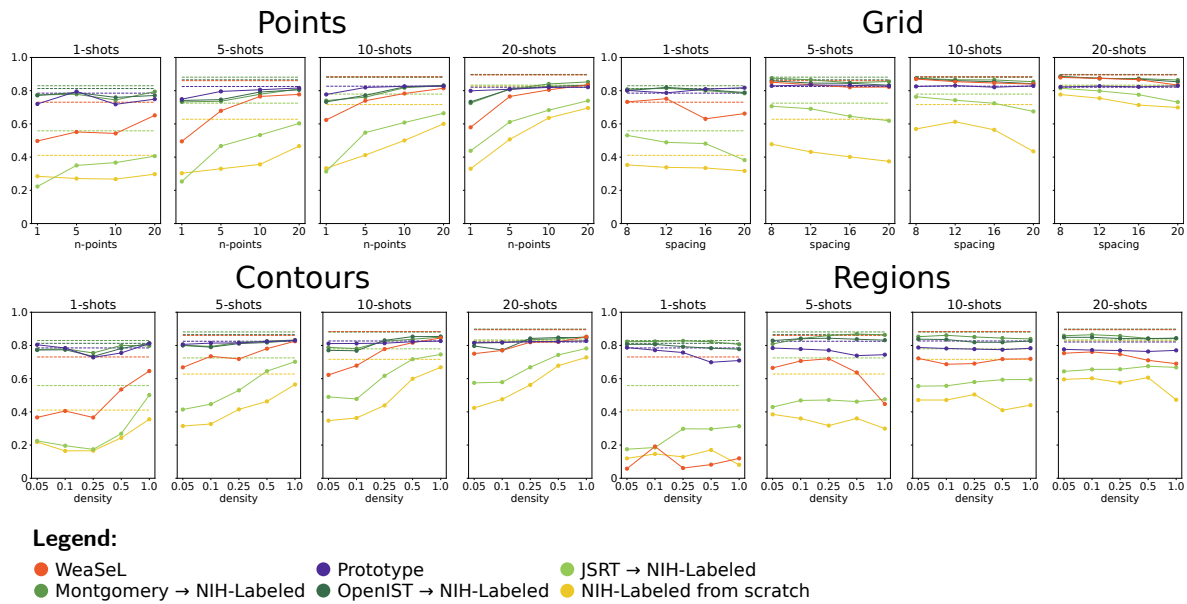


Figure 6.5: Jaccard Score of experiments with NIH-Labeled Lungs Task.

average of the features of labeled pixels to produce the class prototypes, if the embedding function is learned properly, the increase or decrease of labeled pixels will have little effect in the average vector. Increasing this number will only make the mean tend to the average vector of the dense case. Thus, as will be seen in the other experiments, the Prototypical method will mostly have a constant score across different tasks settings.

These results seems to show that, given proper training time, the WeaSeL can exploit the information of different tasks and provide a good starting point for fine-tuning. Additionally, the Prototypical method, albeit being designed to learn an unbiased ensemble function, benefit from similar tasks in the training distribution and seem to assimilate information of these tasks, thus performing better in similar cases. We have

this assumption for the Prototypical method, because, excluding the *JSRT Lungs* task, 5 out of 12 training tasks are a similar CXR Lung task. If uniformly distributed, approximately 40% of the time one of these tasks will be selected for a meta-training epoch.

### 6.1.1.2 Mammography X-Ray Datasets

In the MXR Datasets Tasks (5.1.1 [1.G - 1.H]), we see similar tendencies as in the CXR Datasets. Again, fine-tuning from similar tasks appears as a solid solution, with WeaSeL method, in most cases, obtaining comparable results.

In the *MIAS Breast* task (Figure 6.6), fine-tuning from the *INbreast Breast* task is the best approach. In the more challenging *INbreast Pectoral Muscle* task (Figure 6.7), the pre-training using the *INbreast Breast* task Dataset, provided the best starting weights for fine tuning. Since the objects (Pectoral Muscle) in this target class are small (relative to breast, or lungs, for example), being used to the visual of the data seems to be more relevant to the model than the knowledge of the object in other dataset.

The Prototypical method seems consistent in the *MIAS Breast* task (Figure 6.6), with few variances across different task settings. In the *INbreast Pectoral Muscle* task (Figure 6.7), given the difference of most methods to their dense labeled trained versions, we assume that the combination of the few-shot samples and, subsequently, generated sparse annotations were not informative enough, specially in the *Contour Type* annotation scenario.

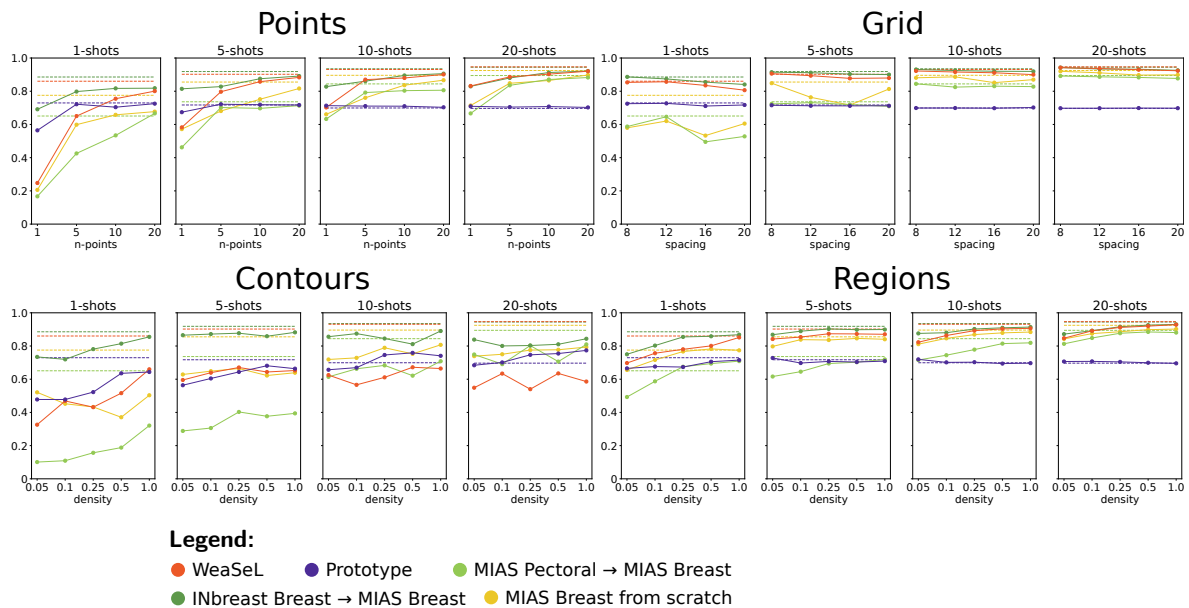


Figure 6.6: Jaccard Score of experiments with MIAS Breast Task.

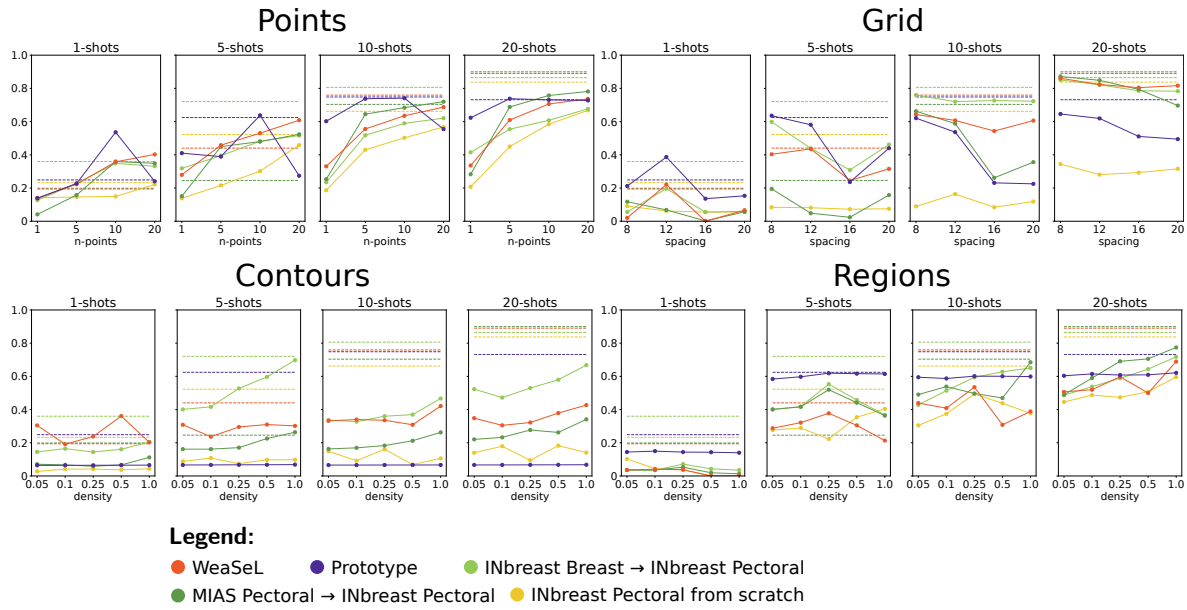


Figure 6.7: Jaccard Score of experiments with INbreast Pectoral Muscle Task.

### 6.1.1.3 Dental X-Ray Datasets

Once more, for the DXR Datasets Tasks (5.1.1 [1.I - 1.J]), we observed the same trend for the evaluated approaches.

This time, without a similar task to fine-tune from, the WeaSeL show to be the best performer, achieving the highest scores in most of the cases, with both dense and sparse annotations. This is observed in the *Panoramic Mandible* task (Figure 6.8) and the *UFBA-UESC Teeth* task (Figure 6.9). Being the most distinct tasks, even the From Scratch baseline, provides satisfying results, and in some cases is the best model.

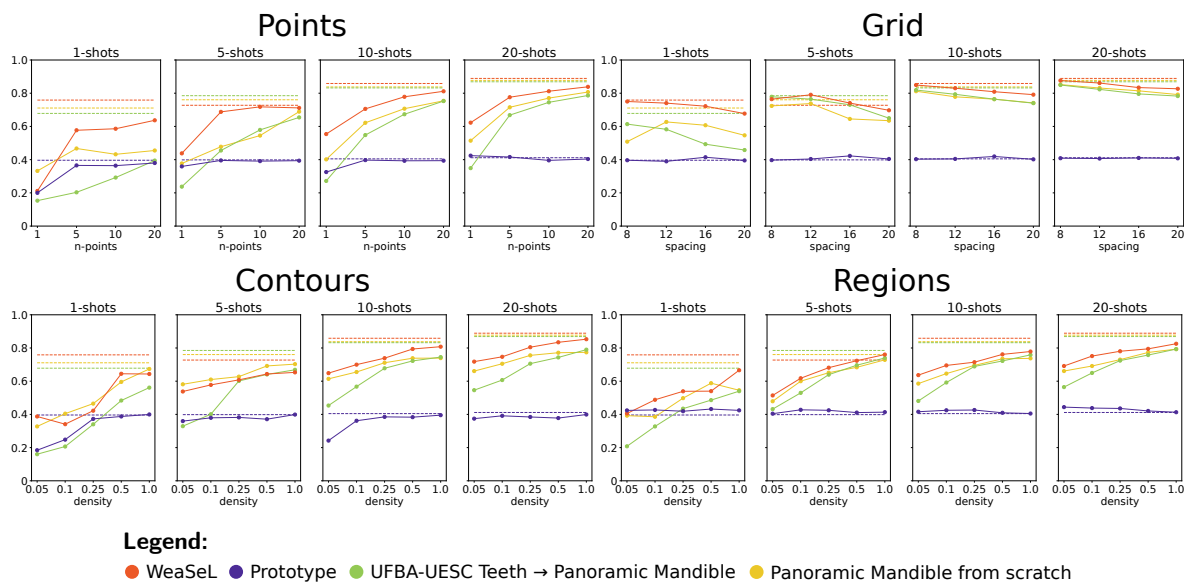


Figure 6.8: Jaccard Score of experiments with Panoramic Mandible Task.

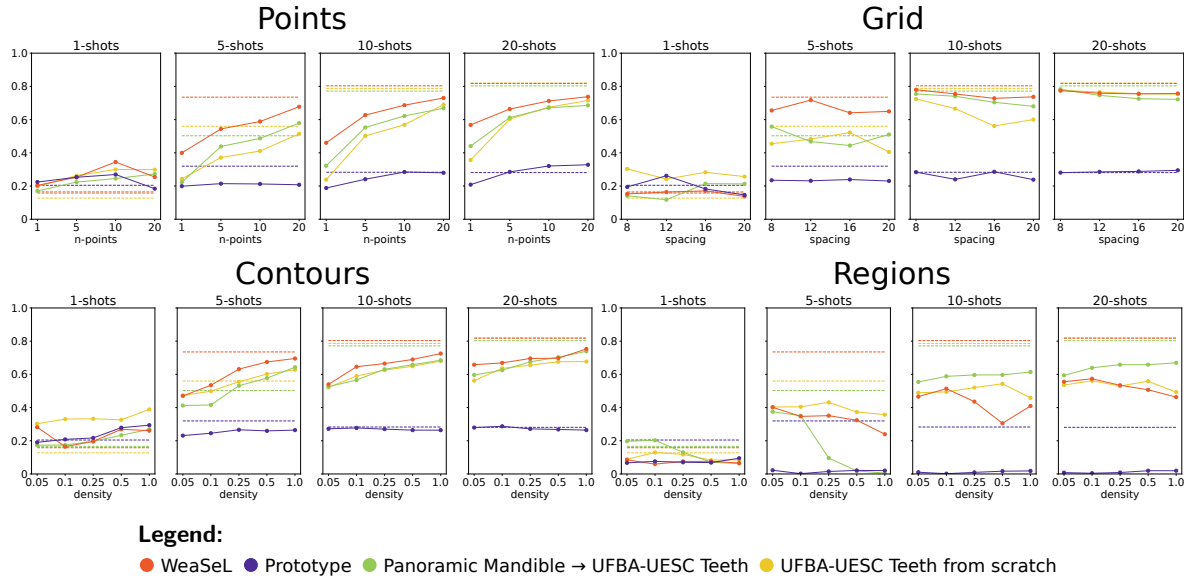


Figure 6.9: Jaccard Score of experiments with UFBA-UESC Teeth Task.

Excluding the target class, only 1 out of 12 tasks is from a DXR dataset. Being the most uncommon type of dataset in our experiments, they have less presence in the meta-training process. This corroborate with our hypotheses that the Prototypical method benefits from similar tasks in training, which explain the relative poor performance of this method in these type of tasks. One can notice a special poor performance for the *Regions Type* annotation for the *UFBA-UESC Teeth* task (Figure 6.9). Our approach to generate these types of annotation appears too strict by only considering *pure regions* (regions within a single class). The resulted superpixels from the SLIC algorithm, most likely included pixels from both classes given characteristics of the target class. Consequently no *pure regions* for the *Teeth* class are generated and selected, which makes the creation of prototypes for this class impossible and explain the approximately zero score.

### 6.1.2 Agricultural Tasks

In general, all Remote Sensing experiments show to be much more difficult than the Medical cases. The higher dimensional input (with three channels) and the higher intraclass variability of most classes, seems to have a great impact in the efficiency of the methods. Overall, no method achieved a Jaccard score above 0.8 in any of the evaluated tasks, not even using dense labels.

For the agricultural tasks, we observe that the WeaSel method consistently outperform the Fine-Tuning and From Scratch baselines, specially in the configurations with the lower amount of data (1-shot tasks). Although being related tasks, the *Coffee* tasks



have an intrinsically domain shift between the different counties in the dataset. The geography of the cities differ, and the methods of plantation can be different. This explains why fine-tuning is not always the best solution. In Figures 6.10 and 6.11 we see the results for the *Montesanto* and *Arceburgo Coffee* task, from the Brazilian Coffee Dataset, while in Figure 6.12 we present the results for the *Orange Orchard* task.

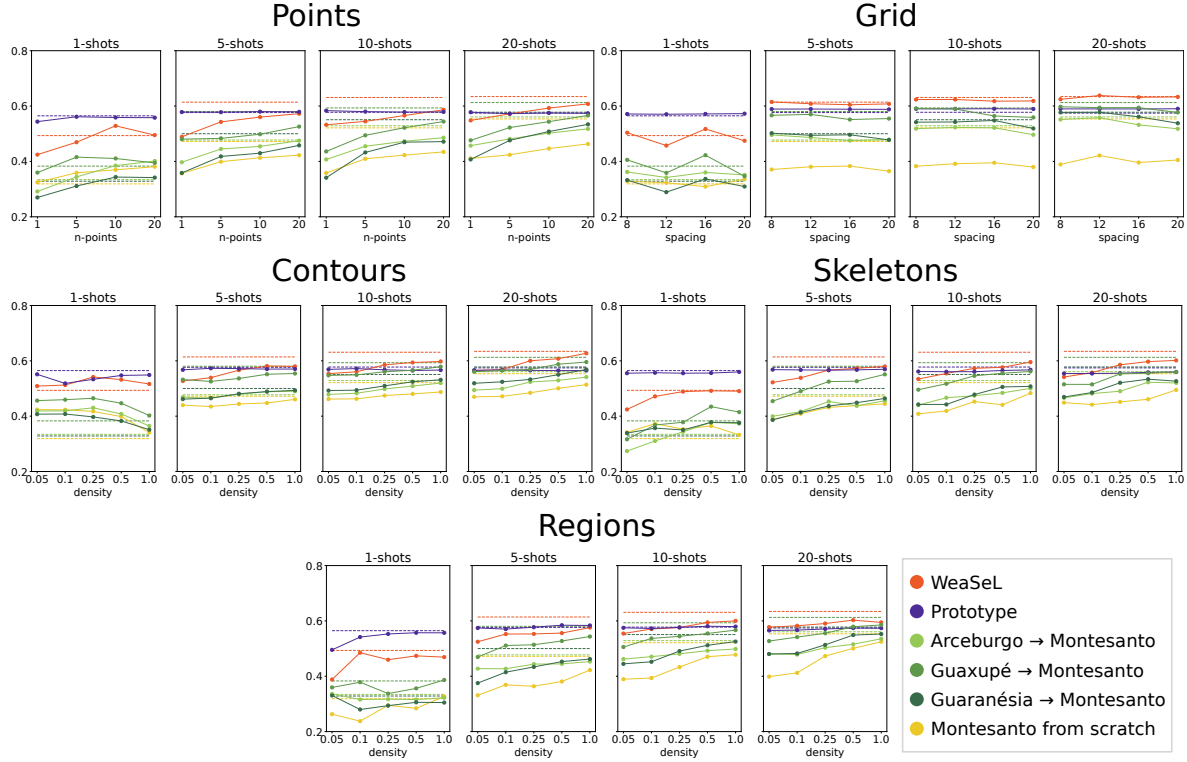


Figure 6.10: Jaccard Score of experiments with Montesanto Coffee Task.

The Prototypical method had consistent results in most tasks. For the *Coffee* tasks, it generally obtained a Jaccard score around 0.6. For the *Orange* task, it revolved around 0.4 Jaccard. In a tendency similar to the Medical experiments, the Prototypical method seems to benefit from related/similar tasks. The *Orange* task was unique in this set of tasks. When choosing *Orange Orchard* as the target task, only the *Coffee* tasks were available to be used in training. The absence of similar tasks appears to affect the Prototypical method results.

### 6.1.3 INRIA Tasks

The INRIA Tasks presents a challenge that although the target class is the same in all tasks, each city have unique characteristics that differentiate them from each other. Also, the intrinsically variability in the target class *Building* increases the difficulty for

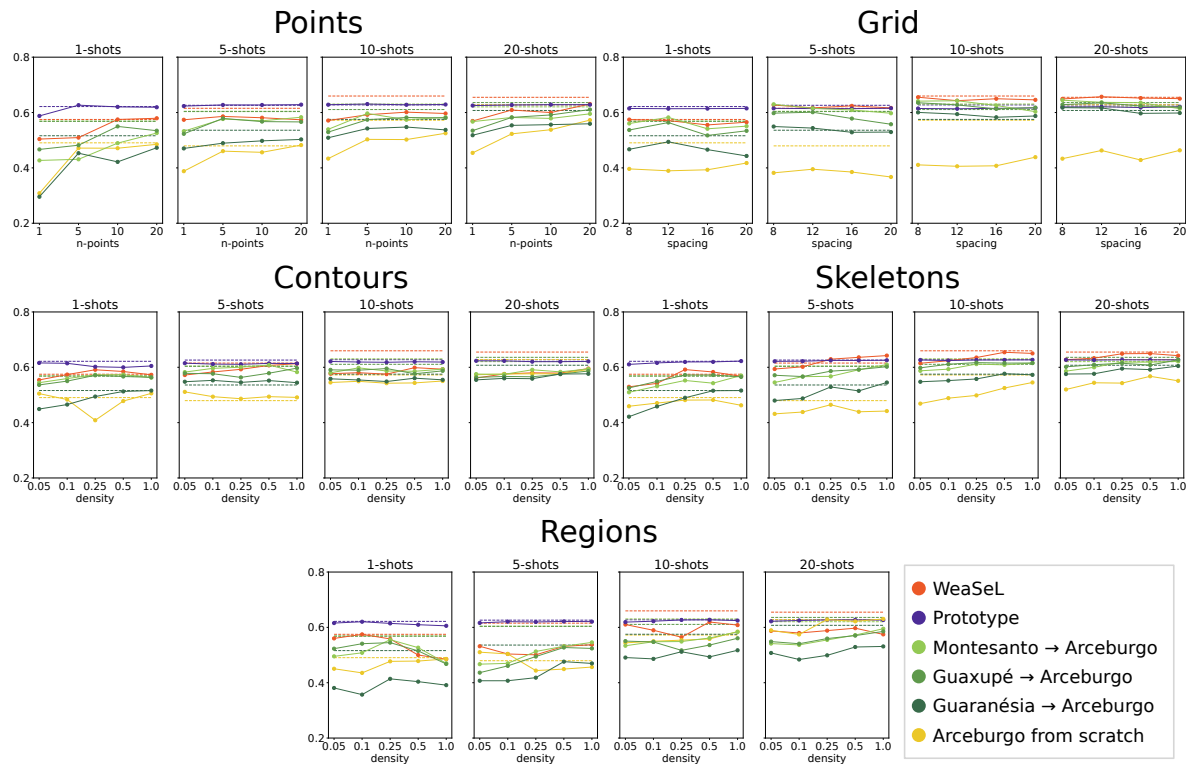


Figure 6.11: Jaccard Score of experiments with Arceburgo Coffee Task.

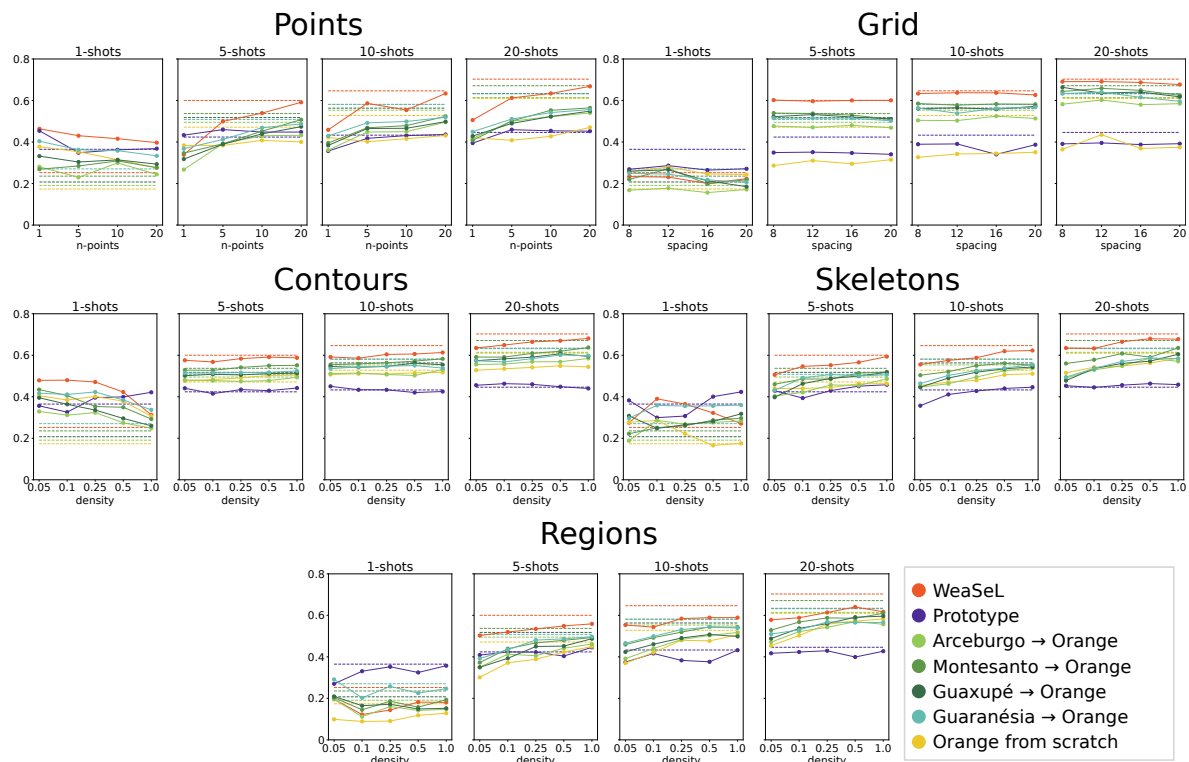


Figure 6.12: Jaccard Score of experiments with Orange Orchard Task.

these tasks. This difficulty is observed when even the baselines, From Scratch and Fine-Tuning, had comparable lower performance than in the Agricultural tasks.

As before, the Prototypical method remained consistent across different tasks and

settings. Given the domain-shifts between tasks, surprisingly, it was the best method in practically all tasks, sometimes with large difference to the second-best method. In Figures 6.13, 6.14, and 6.15, we see the results for the *Austin*, *Kitsap County*, and *Vienna Building* tasks, respectively.

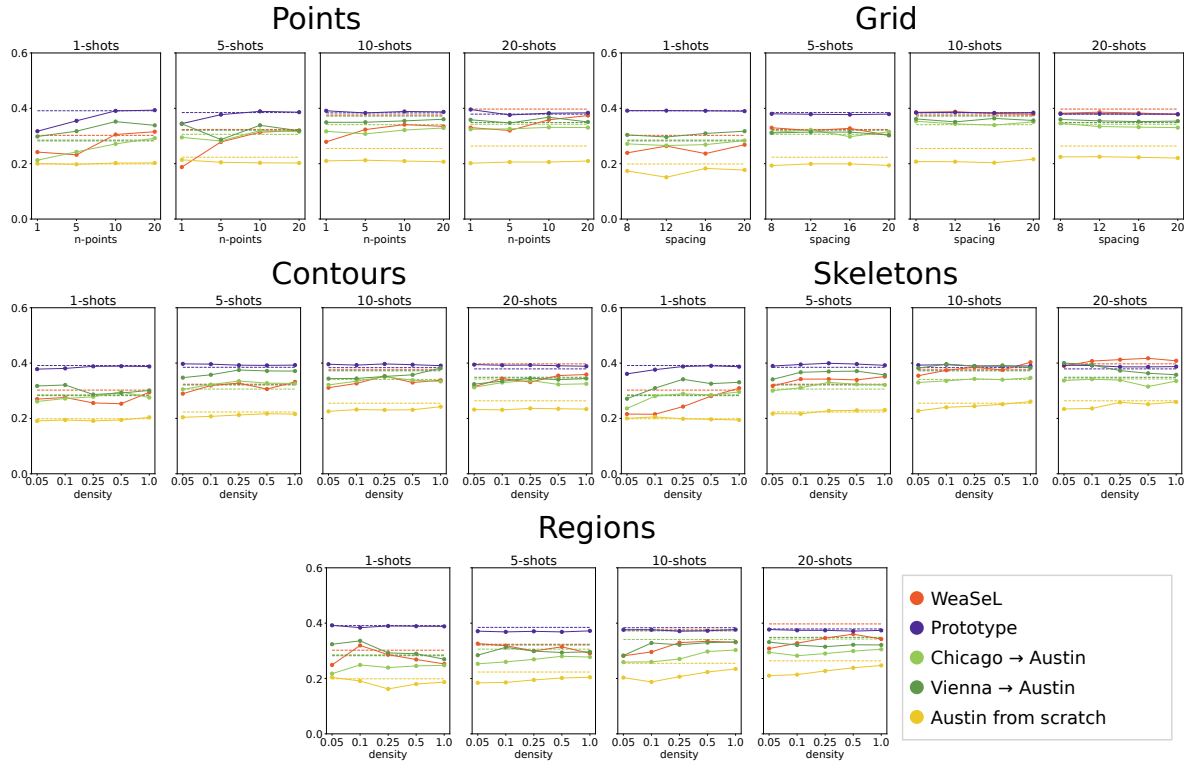


Figure 6.13: Jaccard Score of experiments with Austin Building Task.

In most cases being the second-best, the WeaSeL method outperform the *From Scratch* baseline, and at least one of the fine-tuning baselines in most tasks. Again, with the difference within cities, the WeaSeL method showed to be better as a starting point to fine-tune for a few-shot task than pre-training the model in a highly different dataset.

Different from the Medical Experiments, as mentioned in the section 5.4.2, we used 200 epoch for the meta-training of the WeaSeL and Prototypical methods. We argue that these methods did not fully converge with this number of training epochs, but we choose this number of epochs in order to evaluate a large number of tasks in a limited time. Moreover, there was no hyperparameter optimization. We tested the a configuration of hyperparameters (mentioned in Section 5.4.2) with a single medical experiment, which seemed to converge, and used these same hyperparameters for all tasks. In addition to save time, this choice would permit to see the generalability of our methods performing in different scenarios with the same configuration. That said, as seen in the Medical Experiments (section 6.1.1), we expect that with proper hyperparameters and number of epochs, our methods achieve better results than currently have shown, specially for the ISPRS tasks (Section 6.1.4) that will be show next.

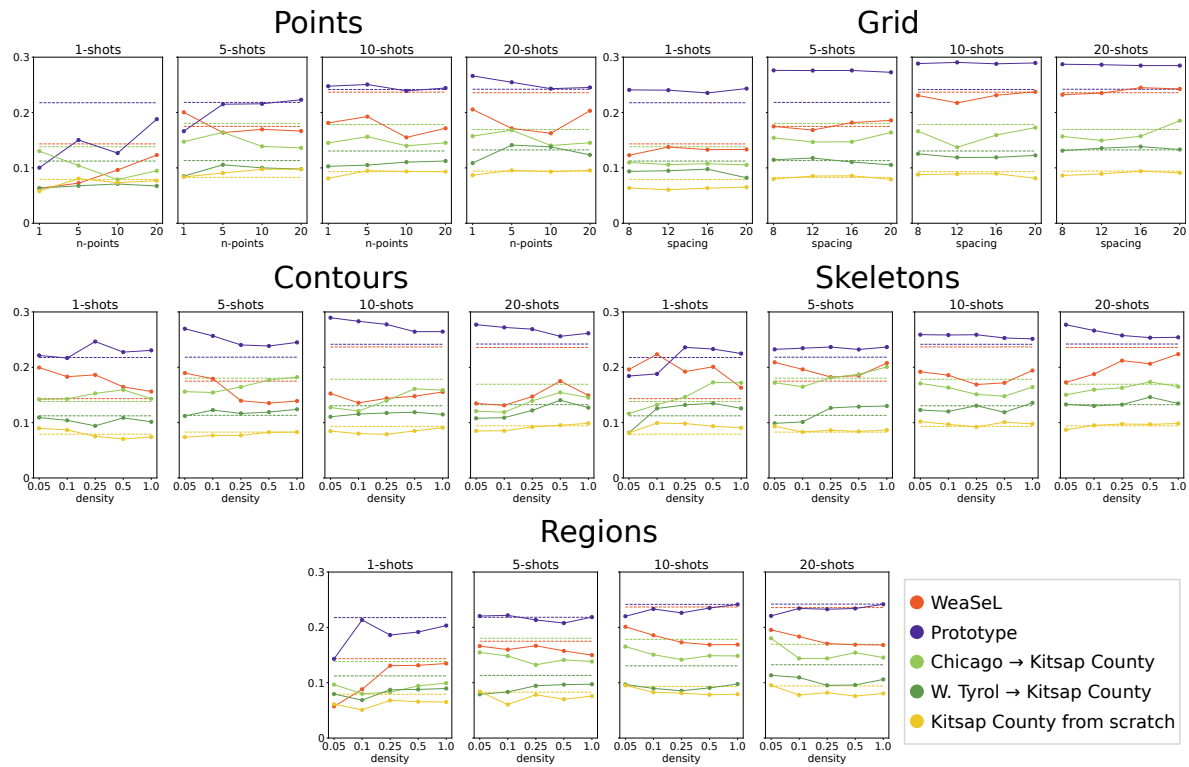


Figure 6.14: Jaccard Score of experiments with Kitsap County Building Task.

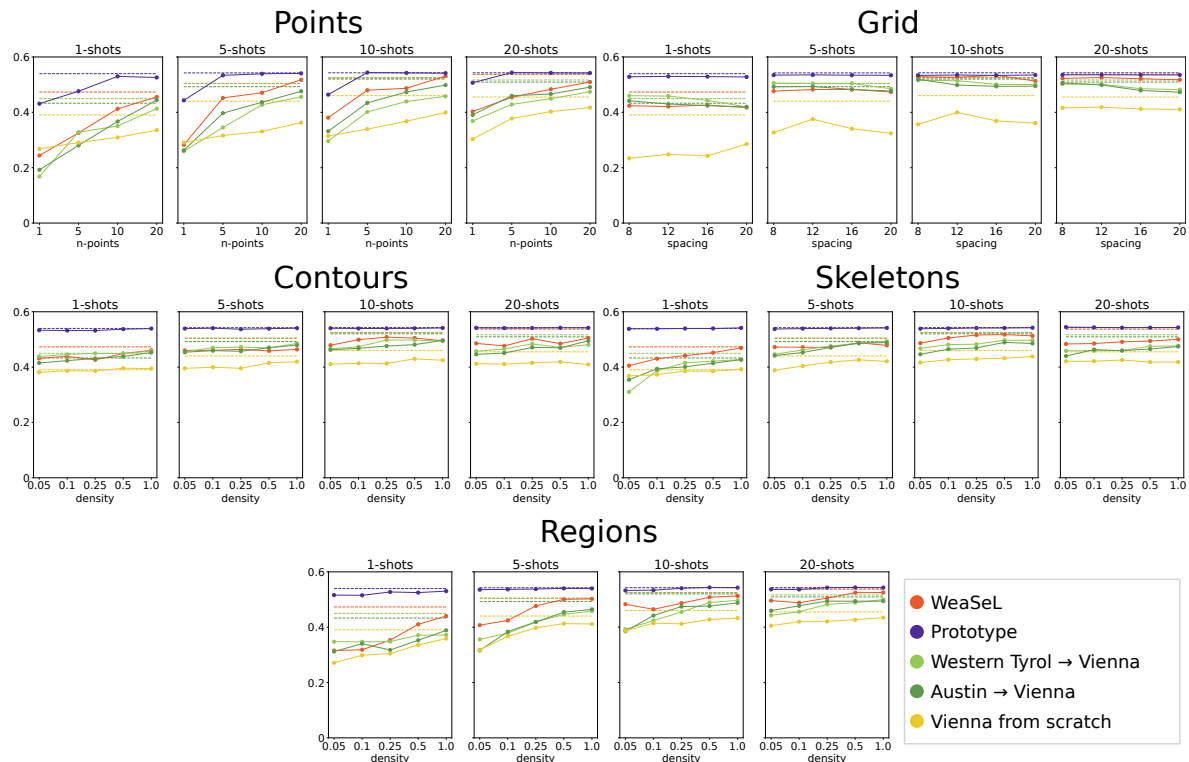


Figure 6.15: Jaccard Score of experiments with Vienna Building Task.

### 6.1.4 ISPRS Tasks

The ISPRS tasks have their own challenges and had the most surprising results. First, the two cities, Potsdam and Vaihingen, used different spectral bands. For Potsdam we used RGB images, and for Vaihingen we used NIRG images. This difference increases the natural domain-shift between different locations. Besides that, the tasks only have one other task with the same target class, which is in a different city. This shift implies that a same class will have different features in each of the cities.

In Figure 6.16 we show the results for the *Vaihingen Road* task, while in Figure 6.17 we present the results for the *Vaihingen Tree* task. Meanwhile, in Figures 6.19 and 6.18 we show the results for the *Potsdam Road* task and *Potsdam Building* task, respectively. Note that for the tasks *Vaihingen Tree* and *Potsdam Building*, only the From Scratch baseline are presented.

Given the difference between tasks, we expected that WeaSeL would be the best approach for the tasks. However, this was not the case for most of the tasks tested. Unexpectedly, the From Scratch baseline, specially with dense annotations, obtained higher Jaccard scores than the other methods, with a few exceptions were it performs worse than or equivalent to the WeaSeL. This is most noticeable for the tasks *Vaihingen Tree* (Figure 6.17) and *Potsdam Building* (Figure 6.18). For the two tasks with Fine-tuning baselines, these baselines were in most cases the best performing, even when the source task was from the other city. Interestingly, for the *Potsdam Roads* task (Figure 6.19), all methods had similar results around the 0.4 Jaccard score independent of sparsity of shot number, with some clear exceptions.

Again, the Prototypical remained consistent for different tasks configurations. The Jaccard score achieved in the tasks were around 0.4 in most of them. Once more, the absence of similar tasks, and the intravariability of some of the classes in the experiments, probably affected the results of the method.

## 6.2 Sparse labels comparison

In this section we show the results for our sparse annotations types comparisons. We choose some tasks, with multiple configurations, to compare the number of user inputs and the scores of our approaches. With these experiments we want to answer the second research question, for at least some of the sparse annotations evaluated in this work. We

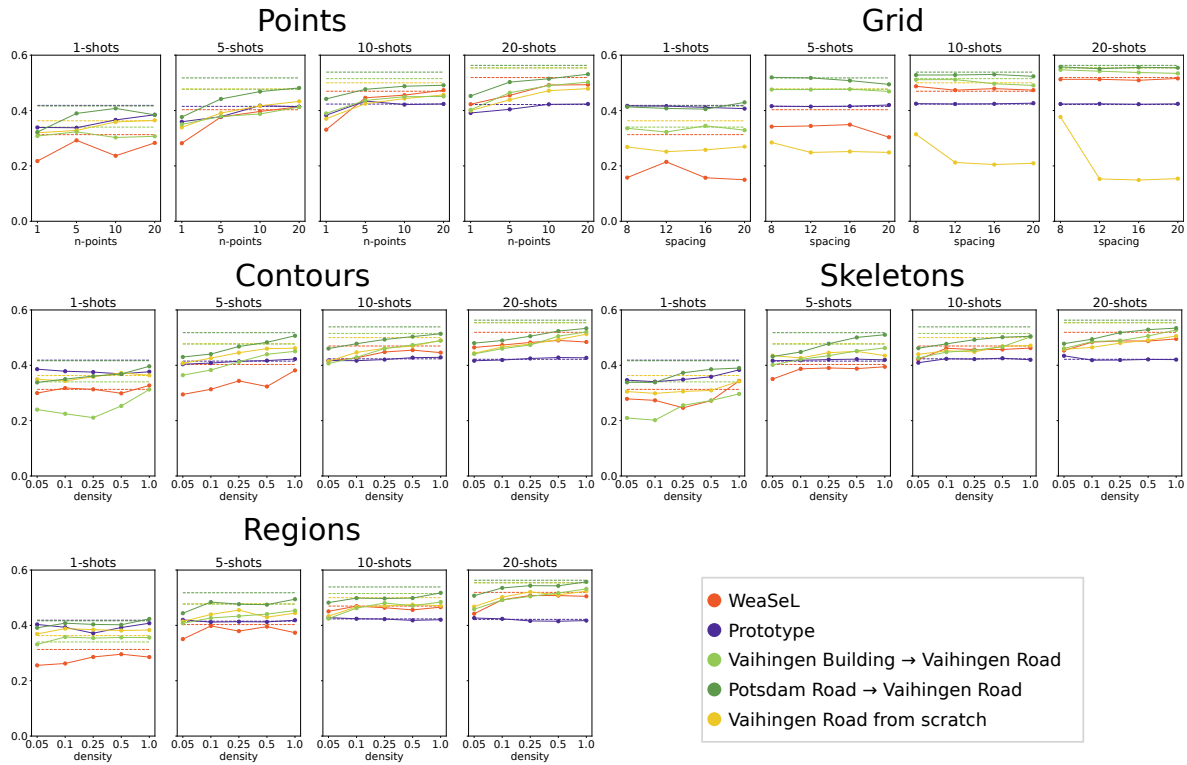


Figure 6.16: Jaccard Score of experiments with Vaihingen Roads Task.

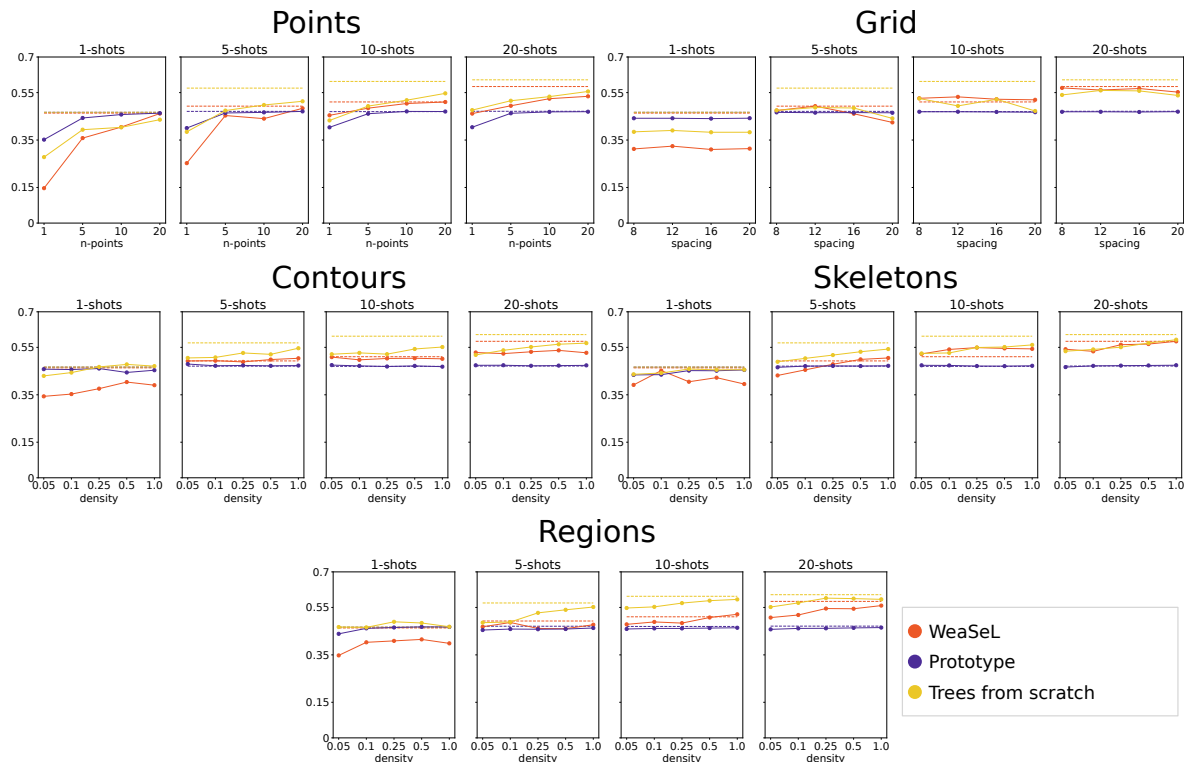


Figure 6.17: Jaccard Score of experiments with Vaihingen Tree Task.

only compared three types of sparse annotations in these experiments: Points, Grid, and Regions.

We defined the number of user inputs for a type of annotation as, the number of

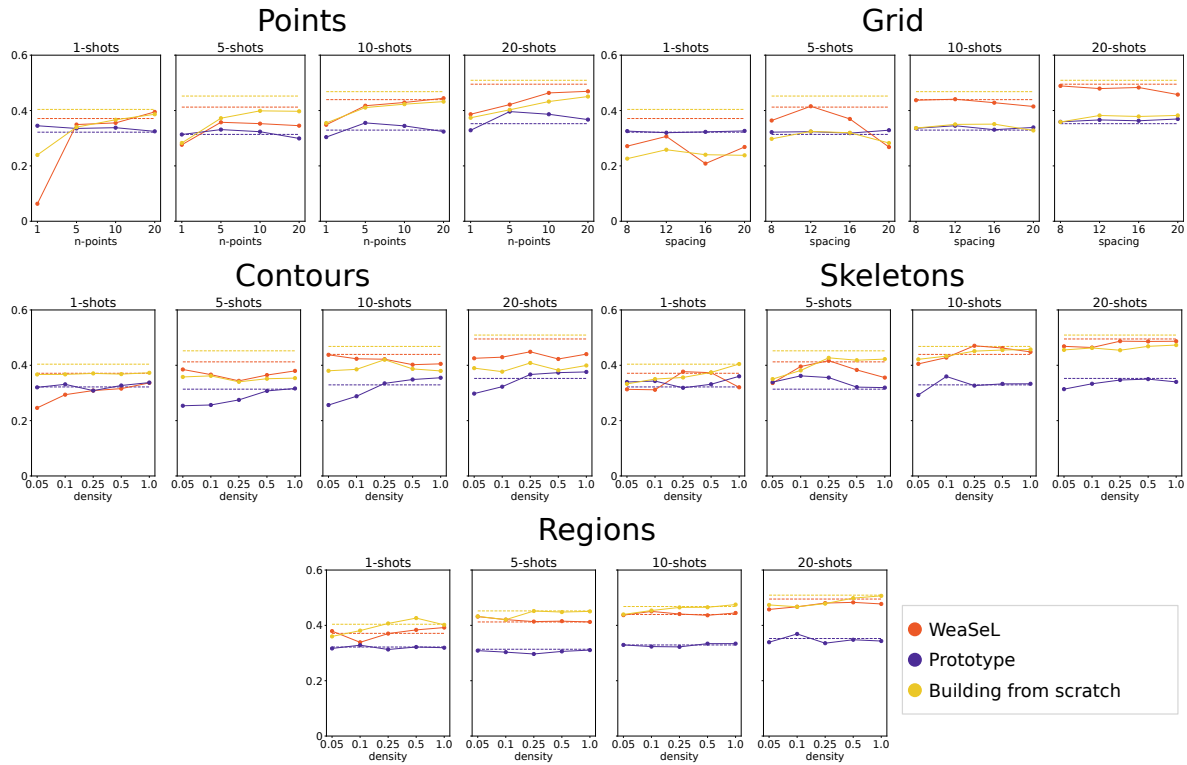


Figure 6.18: Jaccard Score of experiments with Potsdam Buildings Task.

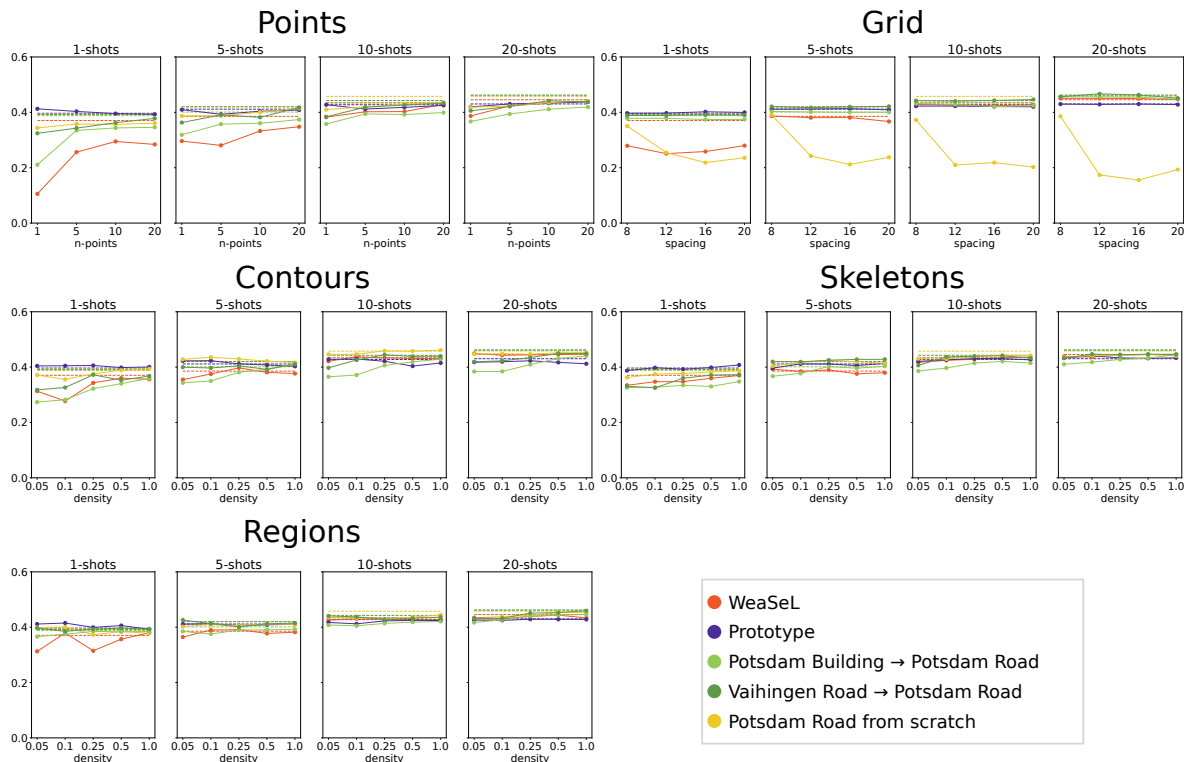


Figure 6.19: Jaccard Score of experiments with Potsdam Roads Task.

interactions an annotator would have to perform in order to sparsely annotate the image using said type. For a single image we defined the number of inputs for a  $n$ -point Points annotation simply as  $2n$ , since the user will select  $n$  positive and  $n$  negative pixels. For

the Grid annotation, we define the number of inputs as the total positive labeled pixels in the grid, since all the pixels are assumed to be negative and the user pick the positive ones. And for the Regions annotation, the number of inputs is defined as the total regions selected, independent of being positive or negative. Once computed the number of inputs for a single image, we sum the values for all images in the support set of the  $k$ -shot task. Finally, we average the number of inputs of the  $k$ -shot across the five folds.

In Figures 6.20, 6.21, 6.22, and 6.23, we present the results for the *JSRT Lungs*, *Montesanto Coffee*, *Austin Buildings*, and *Vaihingen Buildings*, respectively.

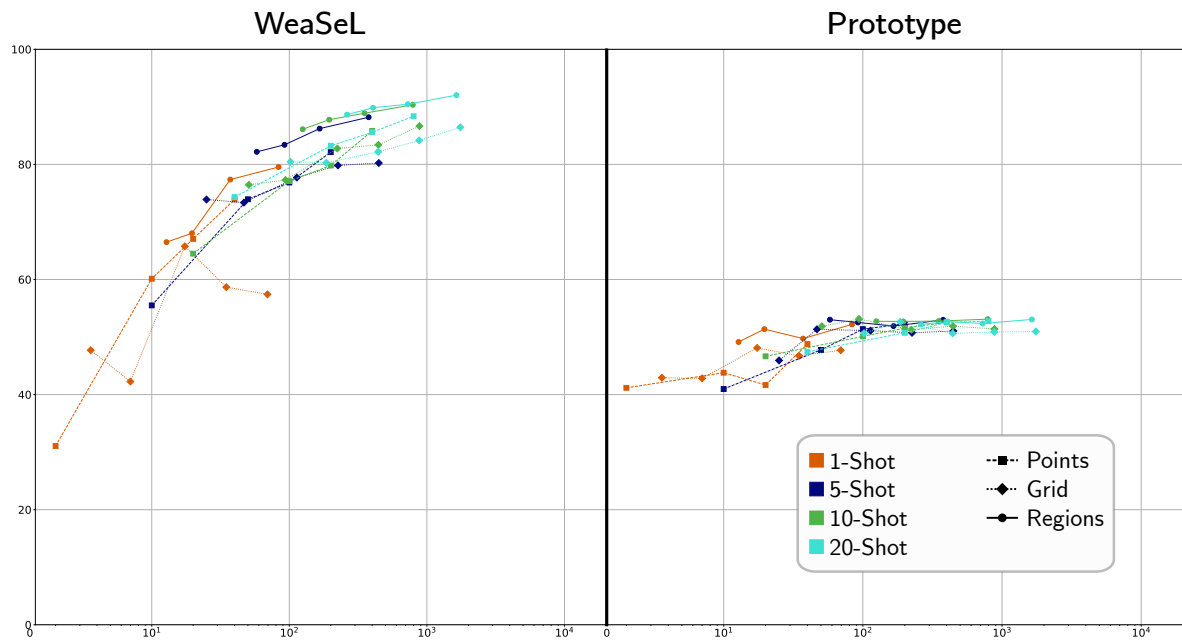


Figure 6.20: Number of user Inputs versus Jaccard Score in the JSRT Lungs Task.

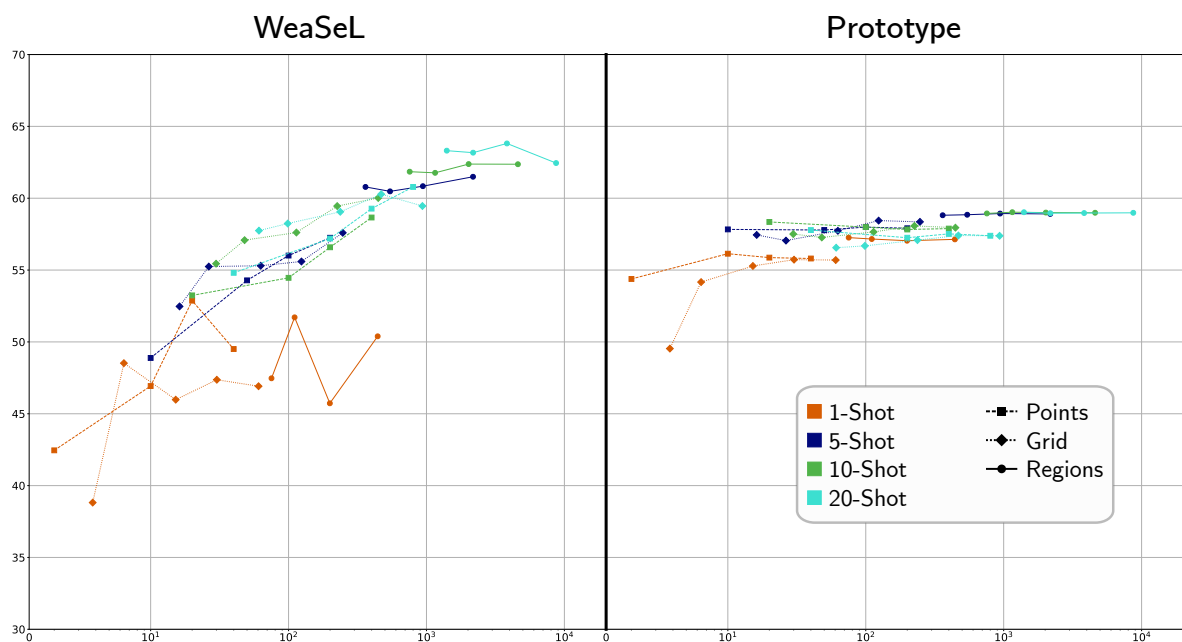


Figure 6.21: Number of user Inputs versus Jaccard Score in the Montesanto Coffee Task.



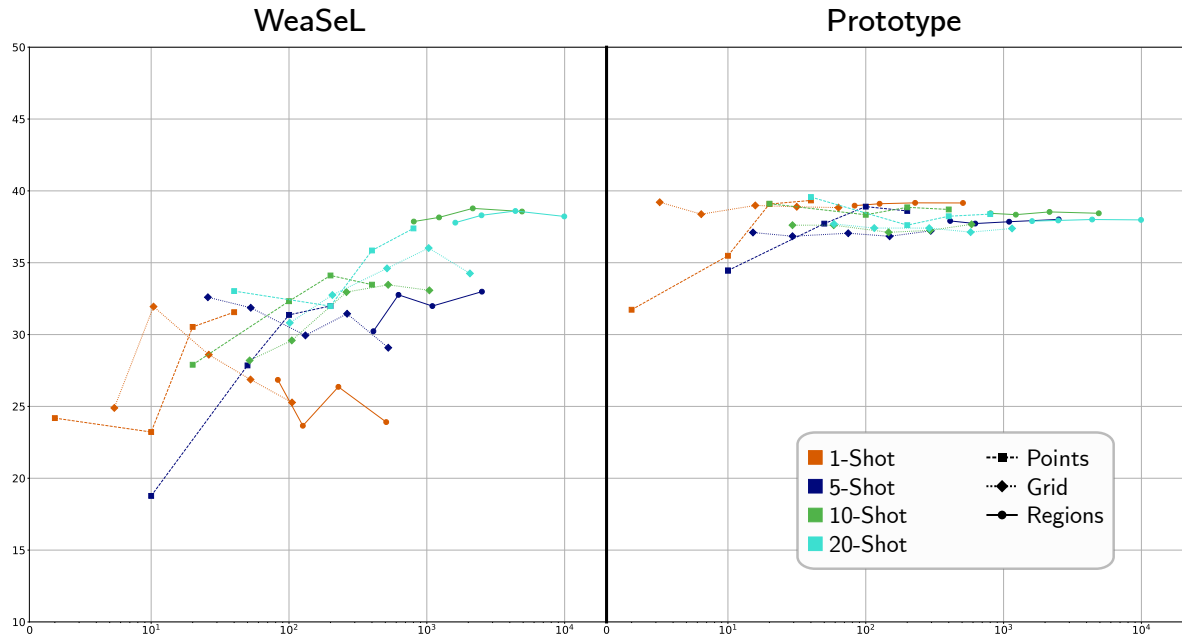


Figure 6.22: Number of user Inputs versus Jaccard Score in the Austin Buildings Task.

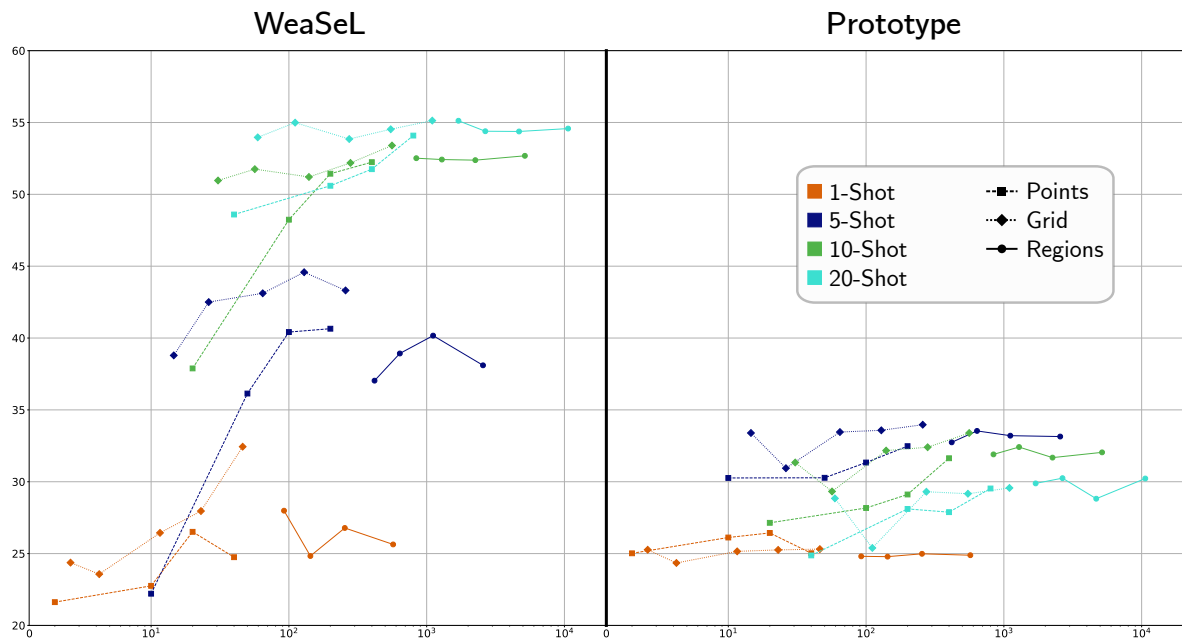


Figure 6.23: Number of user Inputs versus Jaccard Score in the Vaihingen Buildings Task.

We observe that, as seen in the Section 6.1, the WeaSeL method overall performs better with more data information, that increases with the number of user inputs. We also, clearly see how the Prototypical method is almost indifferent to the sparsity and quantity of annotations by having a low deviation score in the presented tasks.

For the *JSRT Lungs* task (Figure 6.20), and Medical tasks, in general, we see that the Grid annotation usually achieve the higher score for the same number of user inputs as the other types of annotation. On the other hand, the Region annotation is commonly the best type of annotation for the Remote Sensing tasks, having higher scores with the same number of inputs. The Points annotation is, at most times, the worse performer.

This was expected, since that with the same number of inputs this type of annotation will have less labeled pixels in total than the other types.

By comparing these results and the ones of the previous section (Section 6.1) we can draw some conclusions. The Grid annotation is a solid annotation type that can lead to good results, and usually is one of the best types for the Medical cases. However, this is the most user consuming type, requiring the larger number of user inputs. The Regions annotation is also a solid option for annotation. When the superpixel segmentation produce clean regions easier to be labeled, this type can make the process of annotation simpler and quicker and produce precise models, specially for Remote Sensing tasks. The Points annotation is the easier to implement and require less from the user. It does not produce the most optimal models, but can lead to comparable results requiring far less inputs. Also this annotation, guarantee a balanced number of pixels samples for each class in training, which can make optimization of the models easier. The other two types of annotations, Contours and Skeletons, appears as valid options as well. The way we designed the process of generating this annotations made it difficult to translate to a countable user input, which is the motive this types are not compared in this section. However, the results presented in Section 6.1, show that both, Contours and Skeletons annotations, are suitable styles, specially Contours in the Medical tasks and Skeletons in Remote Sensing tasks.

## 6.3 WeaSeL Fast Adaptation

In this section we present the results for the adaptation time required by the WeaSeL and a Fine-tuning baseline in some selected tasks.

For this experiments we want to evaluate the number of epochs that the models required to converge in the few-shot task tuning phase. For this we executed a small experiment. We trained the models using the WeaSeL method and using a source dataset. The models were trained for 200 epochs and at epoch 100 and 200 we used the trained weights, at that epoch, as starting points to fine-tune to a few-shot task. Then we observe the Jaccard score for each fine-tune epoch until the epoch 50, where we stopped tuning.

In Figures 6.24 and 6.25 we present the results of the *OpenIST Lungs* and *INbreast Breast* tasks, respectively. Figures 6.26, 6.27 and 6.28 contains the results of *Montesanto Coffee*, *Arceburgo Coffee*, and *Orange Orchard* tasks, in that order. In the graphs we marked with a vertical line the epoch 40. This epoch is where the few-shot tuning of the Remote Sensing tasks was stopped, in the experiments of Section 6.1, and is half the number of epochs used for the few-shot tuning in the Medical tasks in that experiments. Also,

the numbers over the small graphs represent the number of shots and sparse parameter of the annotation type used, for example, in the top-right graph in Figure 6.25 is the 1-shot *INbreast Breast* task with 20-point annotation.

As already discussed in Section 6.1, the number of epochs of pre-training seems to be insufficient to converge the WeaSeL approach. This condition can make our analyzes of these results less conclusive. Regardless, is clear observing the graphs, the instability of the scores in most cases. This indicate that the models did not fully converge to the few-shot task in the epoch span chosen. Also, we observe that the curves of the WeaSeL method and the Fine-tuning baselines have a similar behavior in most cases. These two observations lead us to believe that the *fast adaptation* of the MAML method did not occur in our experiments. This result can be from the lack of convergence in pre-training due to small number of epochs, or this expected feature of the method simply is not possible with our tested cases.

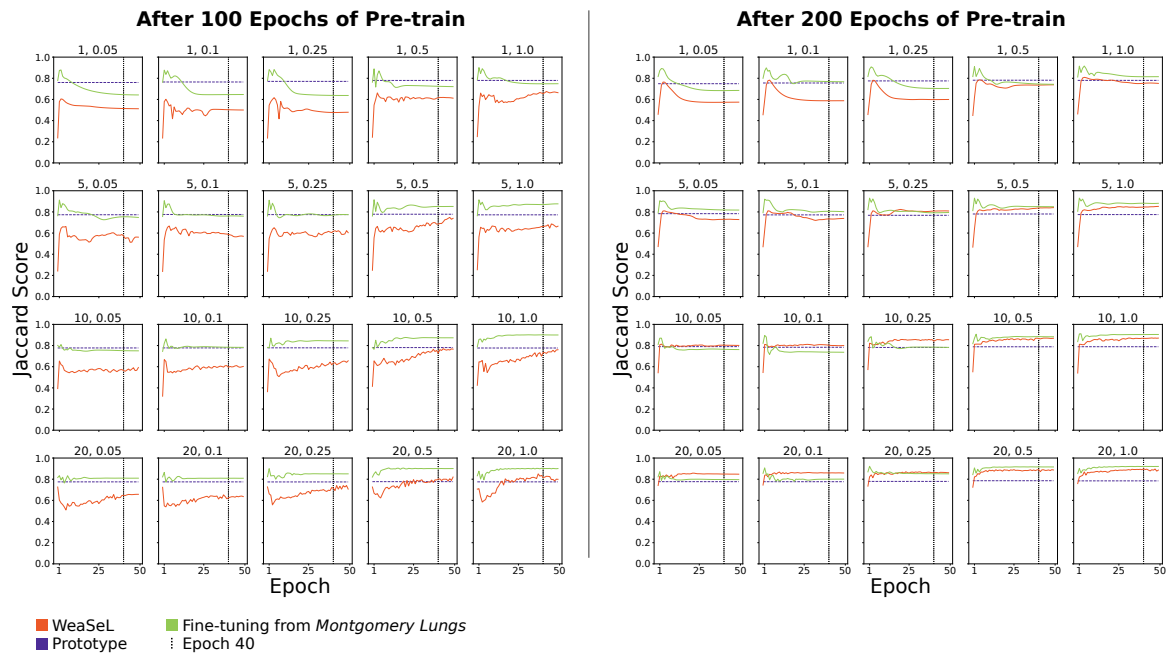


Figure 6.24: Jaccard Score by few-shot fine-tuning Epoch in the OpenIST Lungs task using Contours annotation.

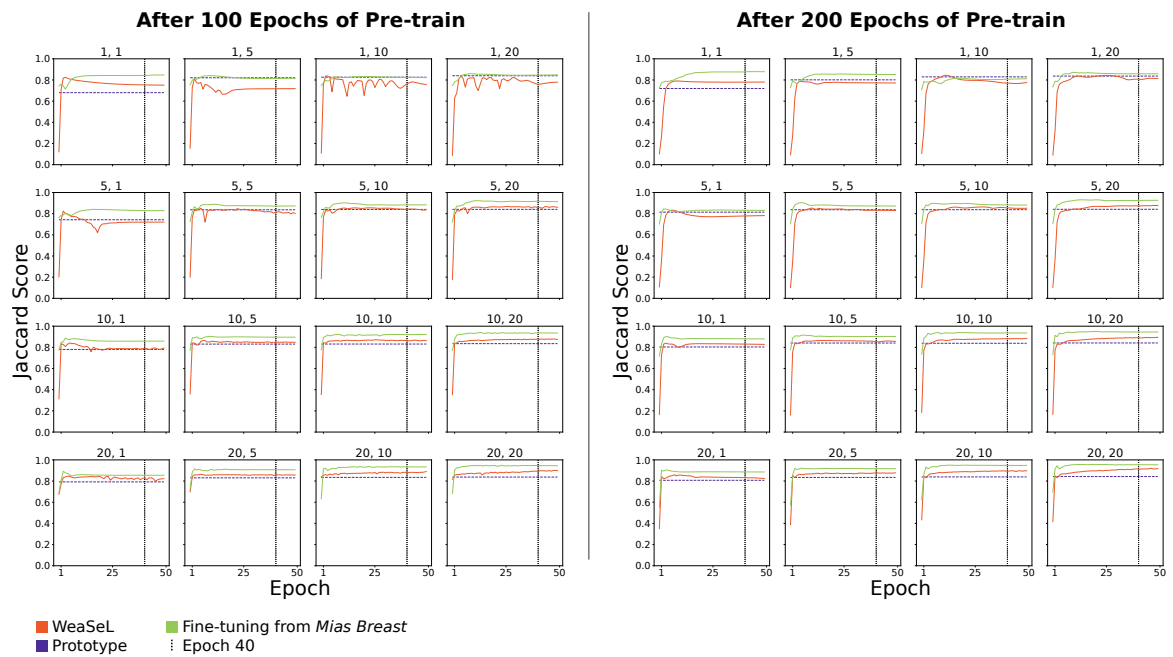


Figure 6.25: Jaccard Score by few-shot fine-tuning Epoch in the INbreast Breast task using Points annotation.

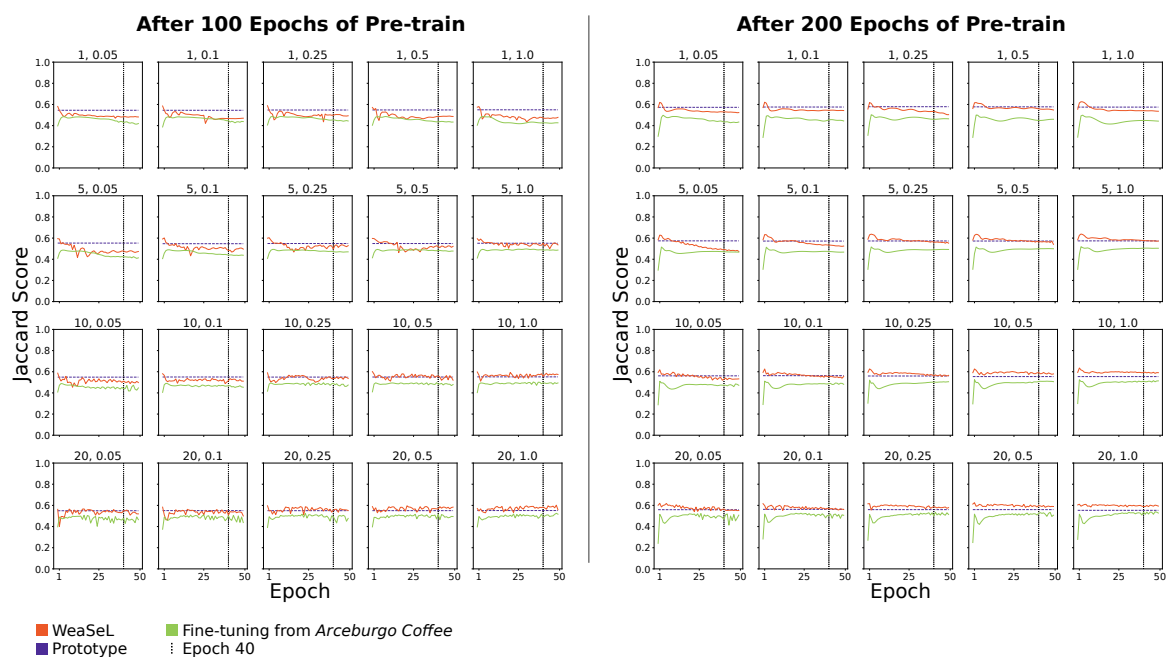


Figure 6.26: Jaccard Score by few-shot fine-tuning Epoch in the Montesanto Coffee task using Skeletons annotation.

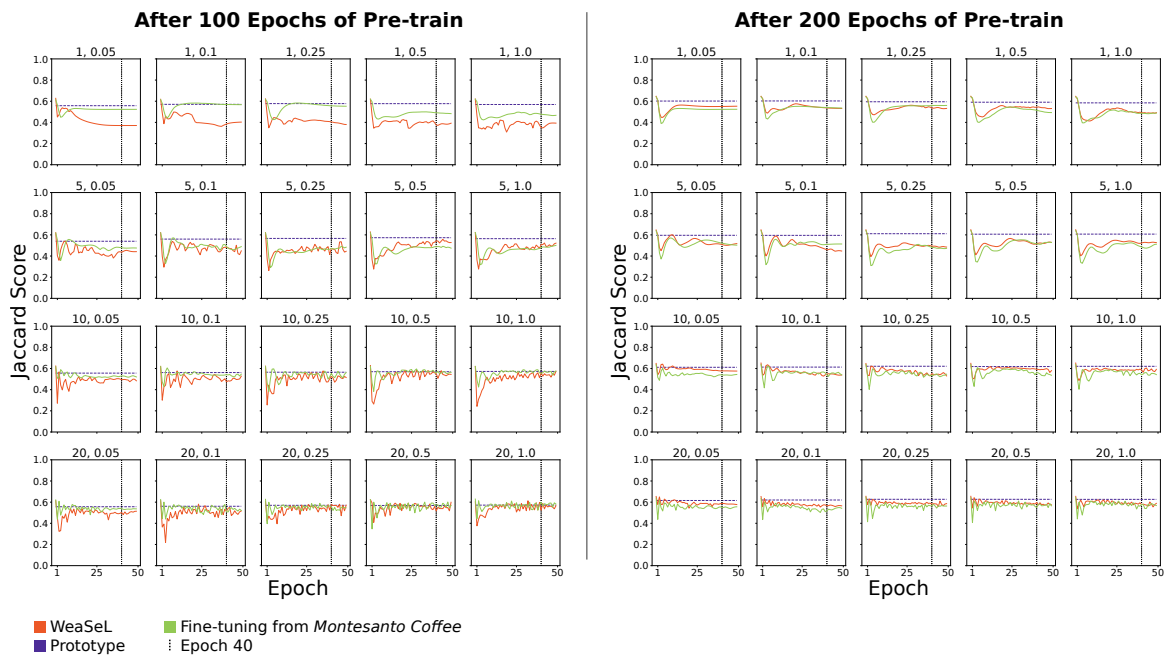


Figure 6.27: Jaccard Score by few-shot fine-tuning Epoch in the Arceburgo Coffee task using Regions annotation.

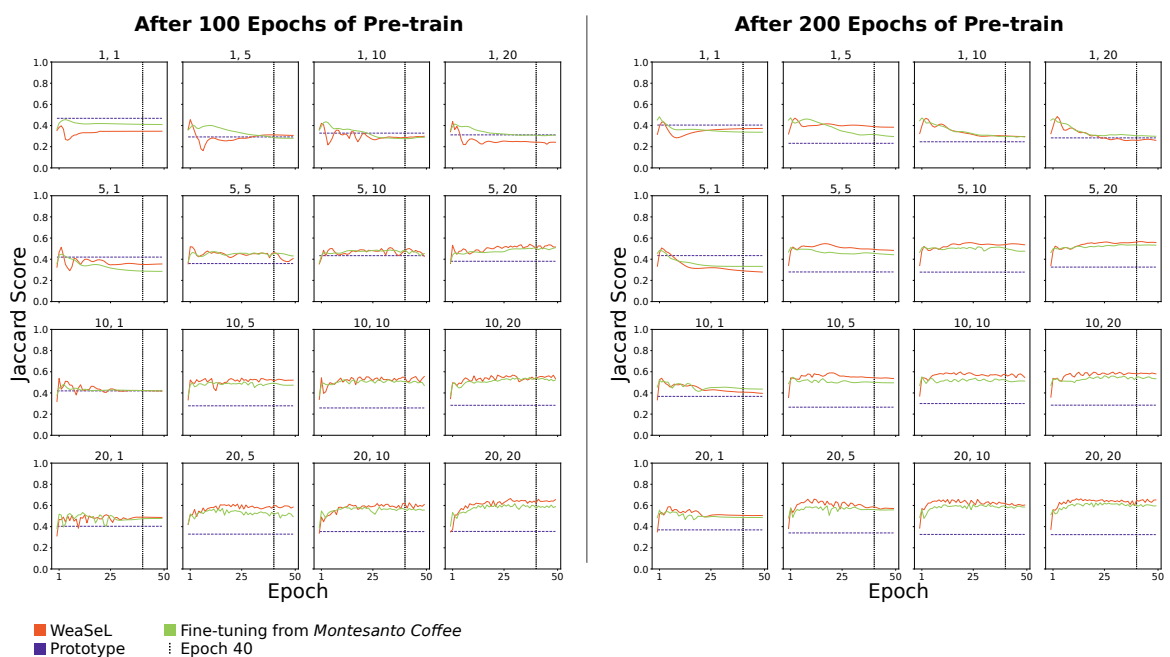


Figure 6.28: Jaccard Score by few-shot fine-tuning Epoch in the Orange Orchard task using Points annotation.

# Chapter 7

## Conclusion and Future Works

In this work we proposed two approaches to the problem of Few-Shot Semantic Segmentation with Sparse Annotations. This is a challenging problem that is still gaining attraction from the scientific community. Our approaches were based on two meta-learning methods originally proposed for Few-Shot Classification problems: the MAML algorithm [Finn et al., 2017] and the Prototypical Networks [Snell et al., 2017].

We evaluated our two methods in a large number of varied experiments to verify their generality. We choose to focus the experiments in two areas that can benefit from the labeling process of few-shot sparse labeled semantic segmentation problems. The two areas, Medical Imaging and Remote Sensing, can have limited data access, thus solving problems with small datasets with few annotations can lead to reveal a new myriad of problems that can be automated in these areas. The experiments were planned in order to clarify three questions: (i) What is the impact of introducing sparse annotations in few-shot scenarios? Can the methods obtain results comparable to using complete dense annotations? (ii) What is the efficiency of different types of annotation? How much data the user have to provide to obtain a useful model? (iii) How different methods converge when training in a few-shot task?

Our first proposed approach, the WeaSeL method, obtained promising results. First, it could achieve scores using sparse annotations equivalent to using dense annotations in many of the tested cases. In special, the WeaSeL method appears to be the most suitable approach to cases where the few-shot task have a significant difference to available training tasks. Moreover, in the experiments, when the evaluated few-shot task did not have a similar or related tasks, in general, the WeaSeL method achieved the higher scores surpassing the Fine-Tuning and From Scratch baselines. Since its difficult to compute the domain shift of datasets in real case scenarios, the WeaSeL method appears as the most appropriate solution for real novel tasks. However, it has some clear problems. First, this method seems to require a large number of epochs to converge. Second, since it depends on tuning in the few-shot task, the model performance will be directed related to the amount of data available, i.e. the number of shots (support size) and the sparsity of annotations. Finally, the WeaSeL method has a high computational cost due to the computation of second-order derivatives in the outer loop, which increases the time

required for training the models. As seen in the third part of experiments (Section 6.3), this method did not seem to provided the *fast adaptation* expected from MAML in our tests, but this could be a consequence of the experiments settings.

Our second proposed approach, the Prototypical method, appears as a more stable solution. The results of this method for few-shot tasks appears to be correlated to the availability of similar tasks in training, as some of our experiments showed. This can reduce the applicability of the method in real cases, as it is not as versatile as the WeaSeL method. One key aspect of the Prototypical method, that can be seen in all experiments (specially in Section 6.2) is that it showed indifference to the amount of data available in the few-shot task. Once trained, the Prototypical method constantly obtained results that almost do not change with the sparsity of the annotation or the number of images labeled. The results, in most cases, are equivalent to using dense annotations. If you want to use a minimal number of user inputs in the dataset creation, the Prototypical method is a viable solution that will provide results close to the dense annotations. Nevertheless, it also has its shortcomings. As already mention, the method seemed to be affected by the absence of similar tasks in the training. Additionally, since it performs an average of feature vectors, if a class have a high pixel variability, it can lead to generating a non-informative class prototype that is worthless for comparison. Finally, as it is now proposed, the model do not use pixel spatial relation in consideration for class inference, which is an important factor in semantic segmentation.

The five evaluated types of annotation — *points*, *grid*, *contours*, *skeletons*, and *regions* — have their own pros and cons. The *grid* annotation is one of the more reliable types and produce some of the best results, but is the most user demanding type. The *region* annotation, seems like an efficient option, but it usefulness is dependent on a competent superpixel segmentation. The *points* annotations is the less user demanding, but it also yields the greater gaps for dense annotation scores. The *contours* and *skeletons* annotations appears as valid options, for Medical and Remote Sensing tasks, respectively, but will require further investigation to confirm their efficiency.

In summary, we evaluated our proposed approaches in multiple tasks, and both proved capable of obtaining equivalent results with sparse and dense annotations. As some of the baselines could sometimes achieve these similar results, perhaps few-shot sparse annotations solutions can benefit, but not necessarily require, that methods consider the sparse aspect of the problem.

For future works, first we want to revise some of the experiments executed. We want to test different hyperparameters for Remote Sensing tasks and use a larger number of epochs of meta-training. Second we want to tackle some of the problems of our methods. For the WeaSeL method, we want to use first order approximations of the derivatives in the outer loop and evaluate how this affect the performance of the model. For the Prototypical method, we want to include some form to the model use the spatial

relation of the pixels when predicting the labels. The study of the *contours* and *regions* type of annotation, better forms to count user inputs with these types, and designing experiments that permit the analysis of efficiency of these annotations, is also planned. Moreover, as next steps we include the adaption of other Few-Shot classification methods to the semantic segmentation problem with sparse annotations, as well as proposing novel methods for this problem.



# Bibliography

- Amir Hossein Abdi, Shohreh Kasaei, and Mojdeh Mehdizadeh. Automatic segmentation of mandible in panoramic x-ray. *Journal of Medical Imaging*, 2(4):044003, 2015.
- R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. doi: 10.1109/TPAMI.2012.120.
- Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.
- Samuel G Armato III, Geoffrey McLennan, Luc Bidaut, Michael F McNitt-Gray, Charles R Meyer, Anthony P Reeves, Binsheng Zhao, Denise R Aberle, Claudia I Henschke, Eric A Hoffman, et al. The lung image database consortium (lidc) and image database resource initiative (idri): a completed reference database of lung nodules on ct scans. *Medical physics*, 38(2):915–931, 2011.
- Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images. *Remote Sensing*, 9(4):368, 2017.
- Wenjia Bai, Hideaki Suzuki, Chen Qin, Giacomo Tarroni, Ozan Oktay, Paul M Matthews, and Daniel Rueckert. Recurrent neural networks for aortic image sequence segmentation with sparse annotations. In *MICCAI*, pages 586–594. Springer, 2018.
- John-Melle Bokhorst, Hans Pinckaers, Peter van Zwam, Iris Nagtegaal, Jeroen van der Laak, and Francesco Ciompi. Learning from sparsely annotated data for semantic segmentation in histopathology images. In *International Conference on Medical Imaging with Deep Learning*, 2018.
- Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 221–230, 2017.
- Jinzheng Cai, Youbao Tang, Le Lu, Adam P Harrison, Ke Yan, Jing Xiao, Lin Yang, and Ronald M Summers. Accurate weakly supervised deep lesion segmentation on ct scans: Self-paced 3d mask generation from recist. *arXiv preprint arXiv:1801.08614*, 2018.

- Yigit B. Can, Krishna Chaitanya, Basil Mustafa, Lisa M. Koch, Ender Konukoglu, and Christian F. Baumgartner. Learning to segment medical images with scribble-supervision alone, 2018.
- Dongcai Cheng, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Fusionnet: Edge aware deep convolutional networks for semantic segmentation of remote sensing harbor images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(12):5769–5783, 2017.
- Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *MICCAI*, pages 424–432. Springer, 2016.
- Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):603–619, 2002.
- Gabriela Csurka and Florent Perronnin. A simple high performance approach to semantic segmentation. In *BMVC*, pages 1–10, 2008.
- Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.
- Nanqing Dong and Eric Xing. Few-shot semantic segmentation with prototype learning. In *BMVC*, volume 3, 2018a.
- Nanqing Dong and Eric P Xing. Domain adaption in one-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 573–588. Springer, 2018b.
- Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.
- Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, pages 9516–9527, 2018.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.
- Tao Hu, Pengwan Yang, Chiliang Zhang, Gang Yu, Yadong Mu, and Cees GM Snoek. Attention-based multi-context guiding for few-shot semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8441–8448, 2019.
- Chengquan Huang, LS Davis, and JRG Townshend. An assessment of support vector machines for land cover classification. *International Journal of remote sensing*, 23(4):725–749, 2002.
- Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pages 487–493, 1999.
- Stefan Jaeger, Sema Candemir, Sameer Antani, Yi-Xiáng J Wáng, Pu-Xuan Lu, and George Thoma. Two public chest x-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative imaging in medicine and surgery*, 4(6):475, 2014.
- Pascal Kaiser, Jan Dirk Wegner, Aurélien Lucchi, Martin Jaggi, Thomas Hofmann, and Konrad Schindler. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*, 55(11):6054–6068, 2017.
- Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–9, 2016.
- Hirokatsu Kataoka, Soma Shirakabe, Yudai Miyashita, Akio Nakamura, Kenji Iwata, and Yutaka Satoh. Semantic change detection with hypermaps. *arXiv preprint arXiv:1604.07513*, 2(4), 2016.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

- Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2927–2936. PMLR, 2018.
- Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *CVPR*, pages 3159–3167, 2016.
- Ivan Lizarazo and Joana Barros. Fuzzy image segmentation for urban land-cover classification. *Photogrammetric Engineering & Remote Sensing*, 76(2):151–162, 2010.
- Agustin Lobo. Image segmentation and discriminant analysis for the identification of land cover units in ecology. *IEEE Transactions on Geoscience and Remote Sensing*, 35(5):1136–1145, 1997.
- Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- Laura Lopez-Fuentes, Claudio Rossi, and Harald Skinnemoen. River segmentation for flood monitoring. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3746–3749. IEEE, 2017.
- David G Lowe et al. Object recognition from local scale-invariant features. In *iccv*, volume 99, pages 1150–1157, 1999.
- 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.
- Damian J Matuszewski and Ida-Maria Sintorn. Minimal annotation training for segmentation of microscopy images. In *ISBI*, pages 387–390. IEEE, 2018.
- Javier A Montoya-Zegarra, Jan D Wegner, Konrad Schindler, et al. Semantic segmentation of aerial images in urban areas with class-specific higher-order cliques. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2:127–133, 2015.
- Nathan Moon, Elizabeth Bullitt, Koen Van Leemput, and Guido Gerig. Automatic brain and tumor segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 372–379. Springer, 2002.
- Inês C Moreira, Igor Amaral, Inês Domingues, António Cardoso, Maria Joao Cardoso, and Jaime S Cardoso. Inbreast: toward a full-field digital mammographic database. *Academic radiology*, 19(2):236–248, 2012.

- Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.
- Keiller Nogueira, William Robson Schwartz, and Jefersson A dos Santos. Coffee crop recognition using multi-scale convolutional neural networks. In *Iberoamerican Congress on Pattern Recognition*, pages 67–74. Springer, 2015.
- Yu-ichi Ohta, Takeo Kanade, and Toshiyuki Sakai. An analysis system for scenes containing objects with substructures. In *Proceedings of the Fourth International Joint Conference on Pattern Recognitions*, pages 752–754, 1978.
- Hugo Oliveira, Virginia Mota, Alexei MC Machado, and Jefersson A dos Santos. From 3d to 2d: Transferring knowledge for rib segmentation in chest x-rays. *Pattern Recognition Letters*, 140:10–17, 2020.
- Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.
- Dzung Pham, Jerry L Prince, Chenyang Xu, and Azar P Dagher. An automated technique for statistical characterization of brain tissues in magnetic resonance imaging. *International journal of pattern recognition and artificial intelligence*, 11(08):1189–1211, 1997.
- Nils Plath, Marc Toussaint, and Shinichi Nakajima. Multi-class image segmentation using conditional random fields and global classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 817–824. ACM, 2009.
- M Prasad, MS Brown, F Abtin, S Vasunilashorn, and JG Goldin. Comparison of scleroderma detection techniques in ct images using supervised and unsupervised methods. *International Journal of Biomedical Engineering and Technology*, 1(4):453–464, 2008.
- Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L. Yuille. Few-shot image recognition by predicting parameters from activations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.
- Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alexei A. Efros, and Sergey Levine. Few-shot segmentation propagation with guided networks. *CoRR*, abs/1806.07373, 2018. URL <http://arxiv.org/abs/1806.07373>.

- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410*, 2017.
- Neeraj Sharma, Amit K Ray, Shiru Sharma, KK Shukla, Satyajit Pradhan, and Lalit M Aggarwal. Segmentation and classification of medical images using texture-primitive features: Application of bam-type artificial neural network. *Journal of medical physics/Association of Medical Physicists of India*, 33(3):119, 2008.
- Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *European Conference on Computer Vision*, pages 852–868. Springer, 2016.
- Junji Shiraishi, Shigehiko Katsuragawa, Junpei Ikezoe, Tsuneo Matsumoto, Takeshi Kobayashi, Ken-ichi Komatsu, Mitate Matsui, Hiroshi Fujita, Yoshie Kodera, and Kunio Doi. Development of a digital image database for chest radiographs with and without a lung nodule: receiver operating characteristic analysis of radiologists’ detection of pulmonary nodules. *American Journal of Roentgenology*, 174(1):71–74, 2000.
- Gil Silva, Luciano Oliveira, and Matheus Pithon. Automatic segmenting teeth in x-ray images: Trends, a novel data set, benchmarking and future perspectives. *Expert Systems with Applications*, 107:15–31, 2018.
- Giorgia Silvestri and Luca Antiga. Stereology as weak supervision for medical image segmentation. 2018.
- Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *CVPR*, June 2020.
- Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30:4077–4087, 2017.
- Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.
- J Suckling et al. The mammographic image analysis society digital mammogram database digital mammography ed ag gale, sm astley, dr dance and ay cairns, 1994.
- Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N Chiang, Zhihao Wu, and Xiaowei Ding. Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. *Medical Image Analysis*, page 101693, 2020.

- Youbao Tang, Yuxing Tang, Jing Xiao, and Ronald M. Summers. Xlsor: A robust and accurate lung segmentor on chest x-rays using criss-cross attention and customized radiorealistic abnormalities generation, 2019.
- Ludvík Tesař, Akinobu Shimizu, Daniel Smutek, Hidefumi Kobatake, and Shigeru Nawano. Medical image analysis of 3d ct images based on extension of haralick texture features. *Computerized Medical Imaging and Graphics*, 32(6):513–520, 2008.
- Michael Treml, José Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, et al. Speeding up semantic segmentation for autonomous driving. In *MLITS, NIPS Workshop*, volume 2, page 7, 2016.
- B. van Ginneken, M.B. Stegmann, and M. Loog. Segmentation of anatomical structures in chest radiographs using supervised methods: a comparative study on a public database. *Medical Image Analysis*, 10(1):19–40, 2006.
- Paul Vernaza and Manmohan Chandraker. Learning random-walk label propagation for weakly-supervised semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7158–7166, 2017.
- Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.
- Guotai Wang, Wenqi Li, Maria A Zuluaga, Rosalind Pratt, Premal A Patel, Michael Aertsen, Tom Doel, Anna L David, Jan Deprest, Sébastien Ourselin, et al. Interactive medical image segmentation using deep learning with image-specific fine tuning. *TMI*, 37(7):1562–1573, 2018a.
- Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng. Panet: Few-shot image semantic segmentation with prototype alignment. In *ICCV*, pages 9197–9206, 2019.
- Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.
- Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7278–7286, 2018b.

- 
- Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. Automated relational meta-learning. *arXiv preprint arXiv:2001.00745*, 2020.
- X. Zhang, Y. Wei, Y. Yang, and T. S. Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. *IEEE Transactions on Cybernetics*, 50(9):3855–3865, 2020.
- Zhenxi Zhang, Jie Li, Zhusi Zhong, Zhicheng Jiao, and Xinbo Gao. A sparse annotation strategy based on attention-guided active learning for 3d medical image segmentation. *arXiv preprint arXiv:1906.07367*, 2019.
- Haidong Zhu, Jialin Shi, and Ji Wu. Pick-and-learn: Automatic quality evaluation for noisy-labeled image segmentation. In *MICCAI*, pages 576–584. Springer, 2019.



# Appendix A

## Additional results for the Section 6.1

In this Appendix, we present extra results for the Section 6.1, that were omitted for brevity.

### A.1 Extra Medical Tasks

This section include the results of four omitted tasks of the Medical Experiments. These tasks are: *JSRT Clavicles* (Figure A.1), *Shenzhen Lungs* (Figure A.2), *MIAS Pectoral Muscle* (Figure A.3), and *INbreast Breast* (Figure A.4).

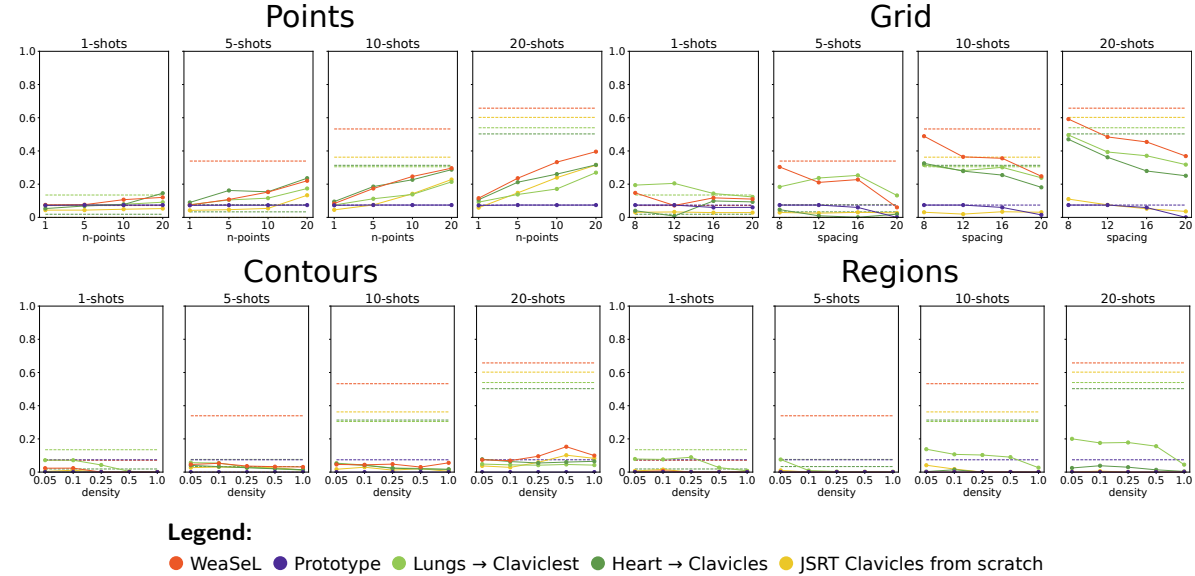


Figure A.1: Jaccard Score of experiments with JSRT Clavicles Task.

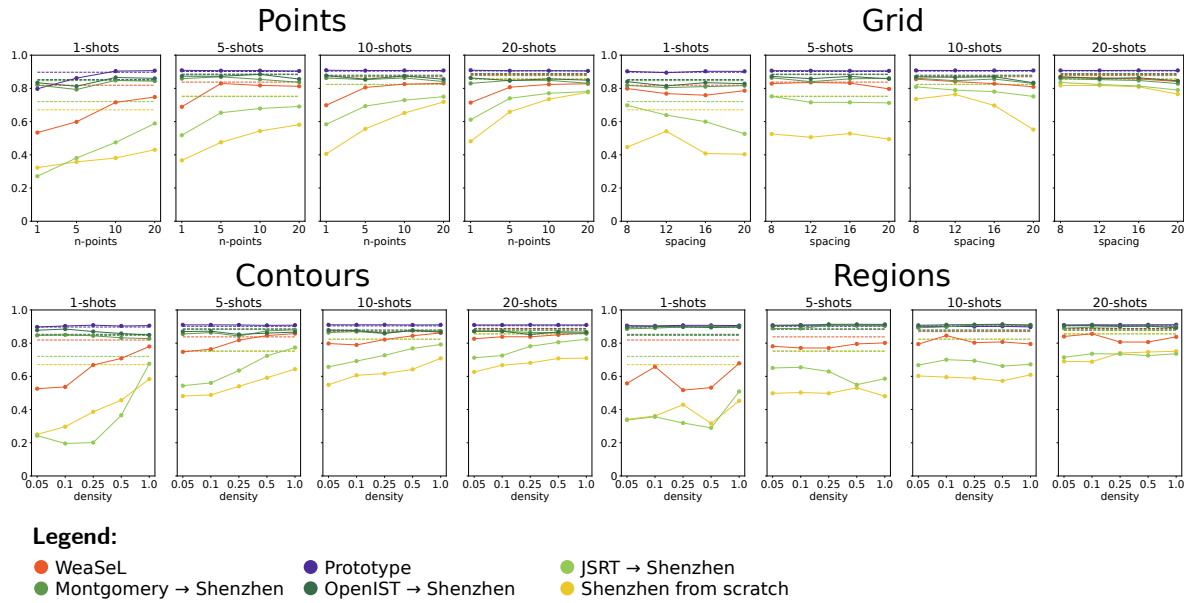


Figure A.2: Jaccard Score of experiments with Shenzhen Lungs Task.

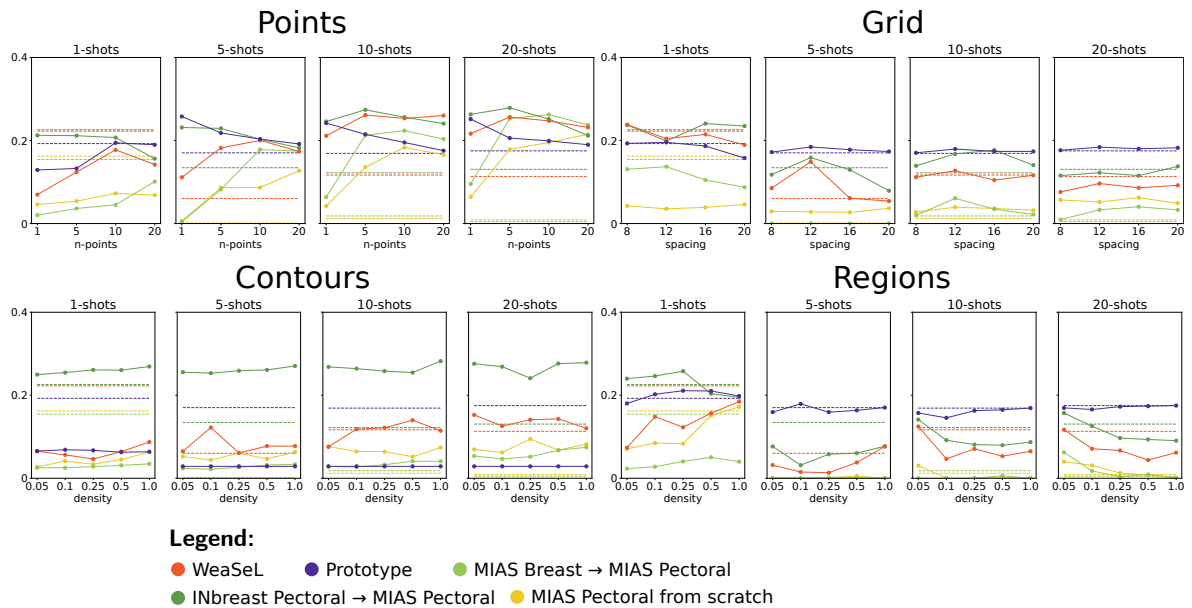


Figure A.3: Jaccard Score of experiments with MIAS Pectoral Muscle Task.

## A.2 Extra Remote Sensing Tasks

This section include the results of eight omitted tasks of the Remote Sensing Experiments. Two tasks from the Brazilian Coffee dataset: *Guaxupé Coffee* (Figure A.5), and *Guaranésia Coffee* (Figure A.6). Two tasks from the INRIA dataset: *Chicago Building* (Figure A.7), and *Western Tyrol Building* (Figure A.8). And four tasks from the ISPRS datasets: *Vaihingen Building* (Figure A.9), *Vaihingen Grass* (Figure A.10), *Potsdam Grass* (Figure A.11), and *Potsdam Tree* (Figure A.12).

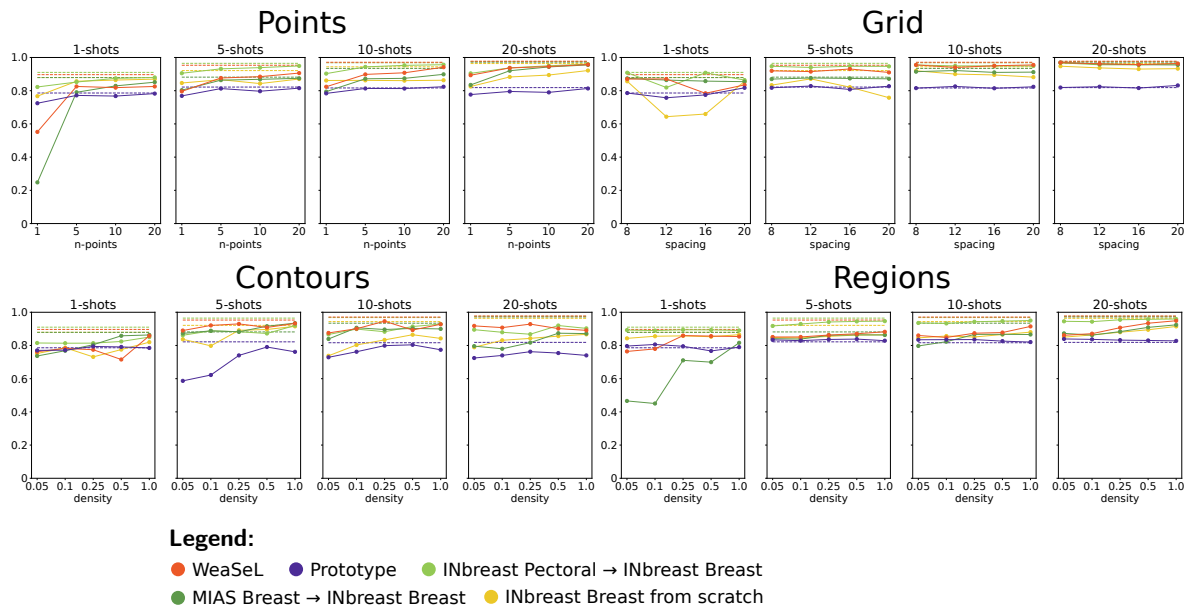


Figure A.4: Jaccard Score of experiments with INbreast Breast Task.

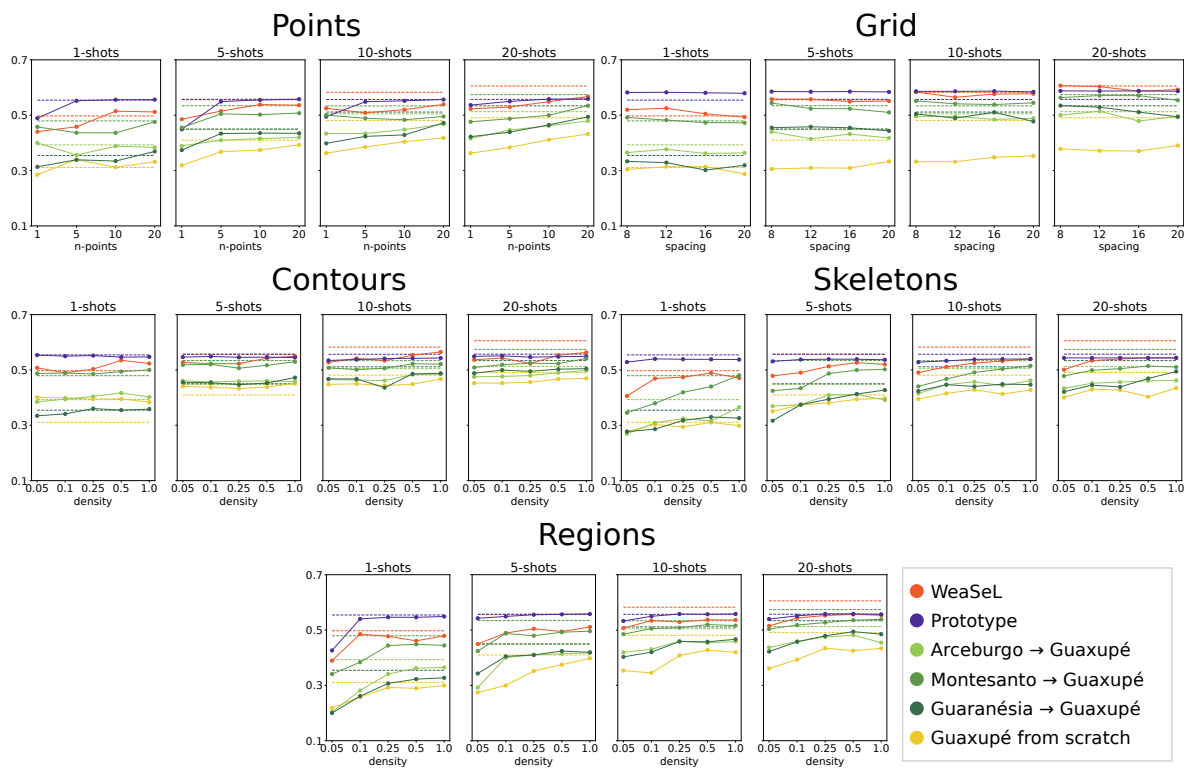


Figure A.5: Jaccard Score of experiments with Brazilian Coffee: Guaxupé Coffee Task.

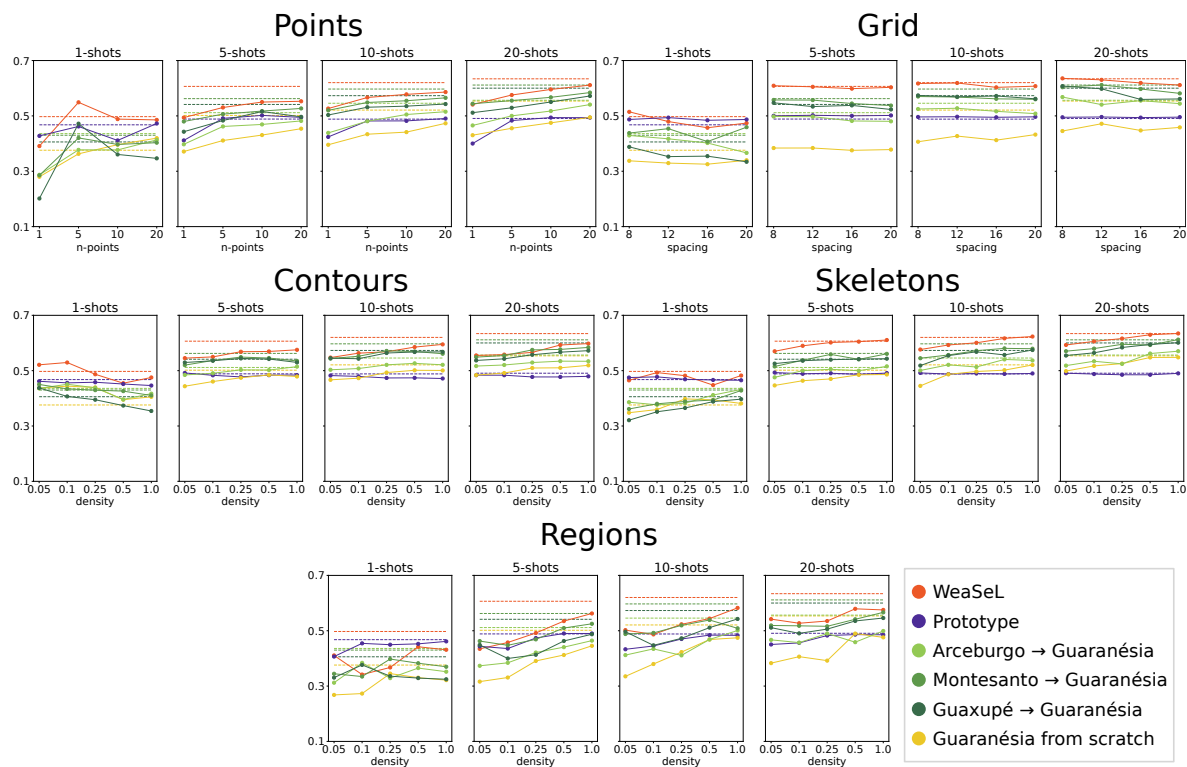


Figure A.6: Jaccard Score of experiments with Brazilian Coffee: Guaraniá Coffee Task.

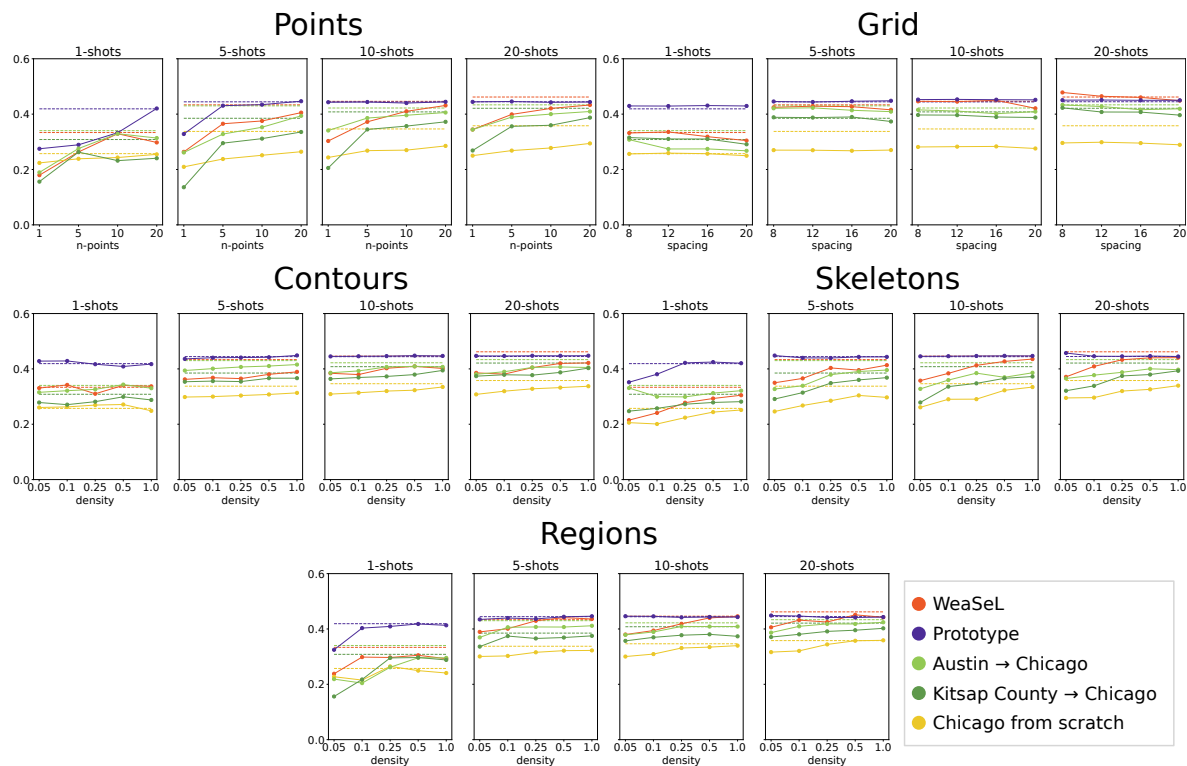


Figure A.7: Jaccard Score of experiments with INRIA: Chicago Building Task.

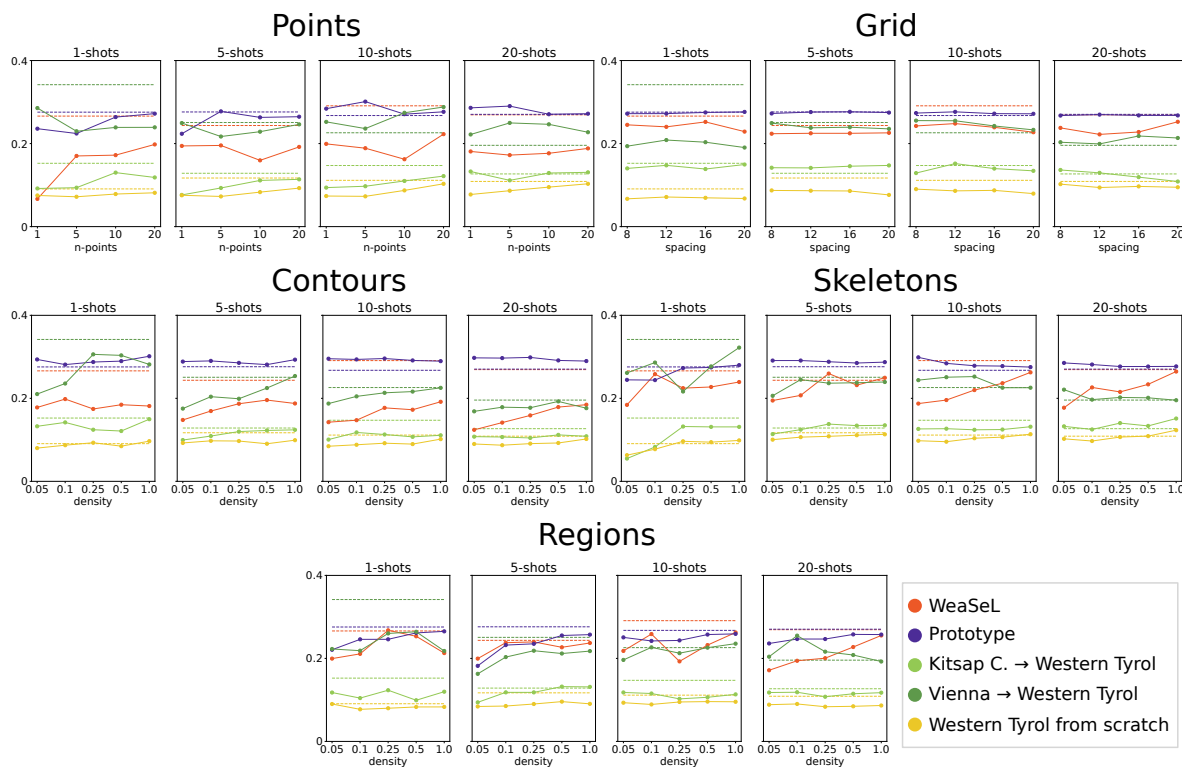


Figure A.8: Jaccard Score of experiments with INRIA: Western Tyrol Building Task.

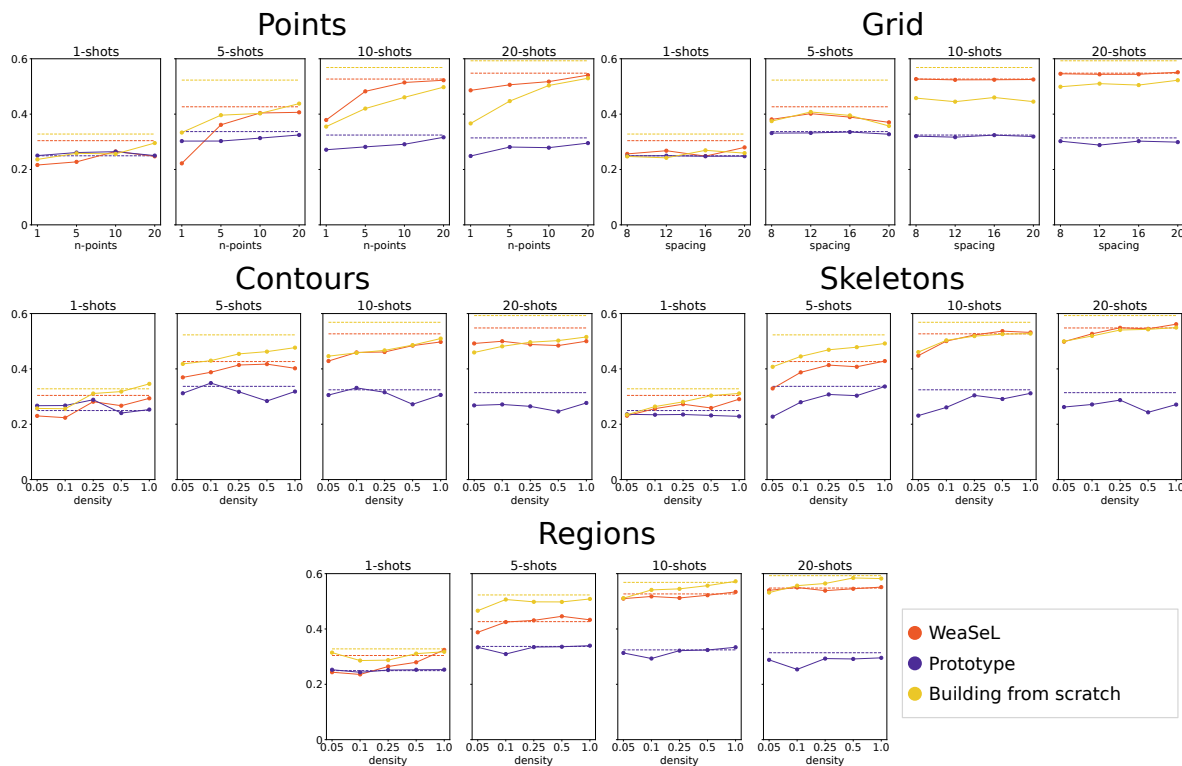


Figure A.9: Jaccard Score of experiments with ISPRS: Vaihingen Building Task.

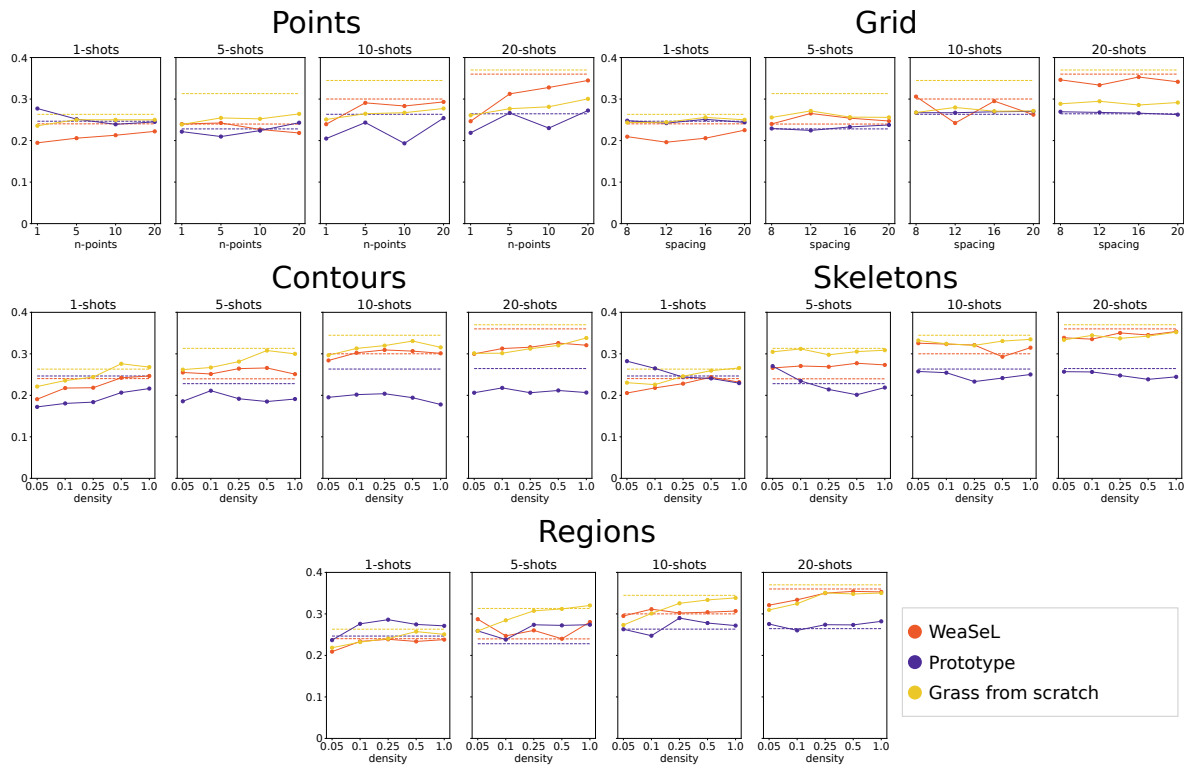


Figure A.10: Jaccard Score of experiments with ISPRS: Vaihingen Grass Task.

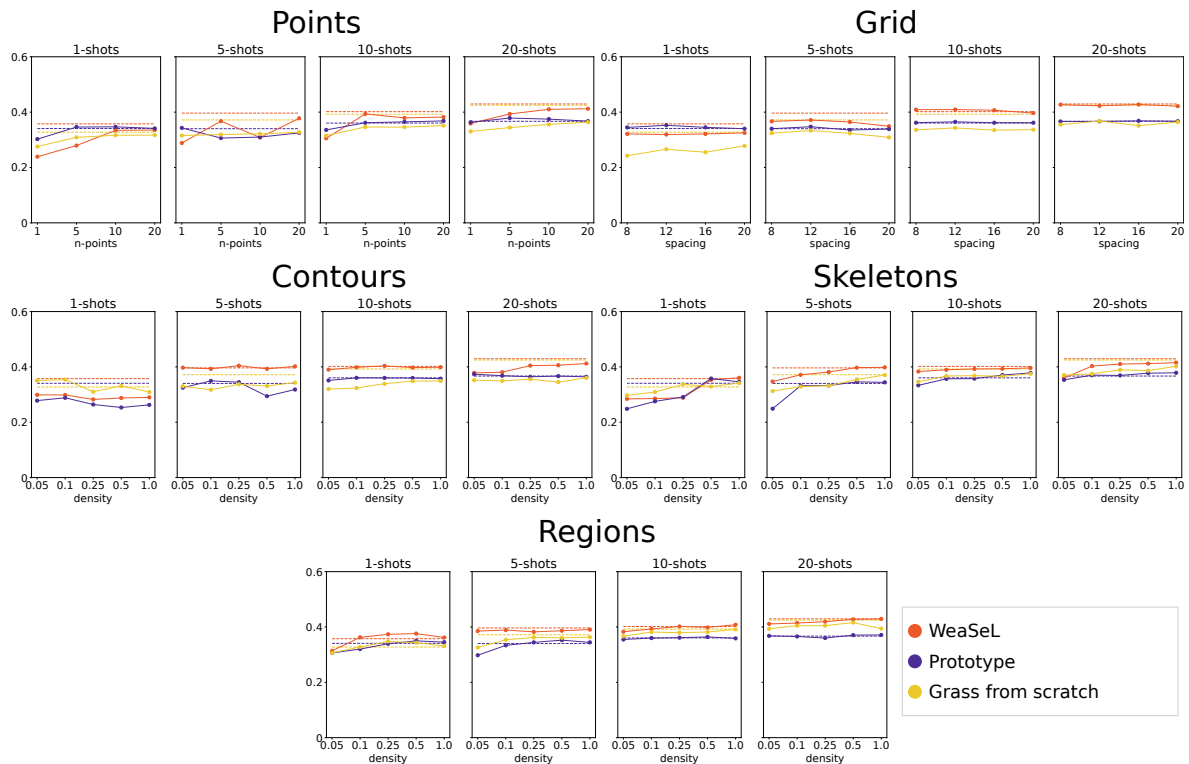


Figure A.11: Jaccard Score of experiments with ISPRS: Potsdam Grass Task.

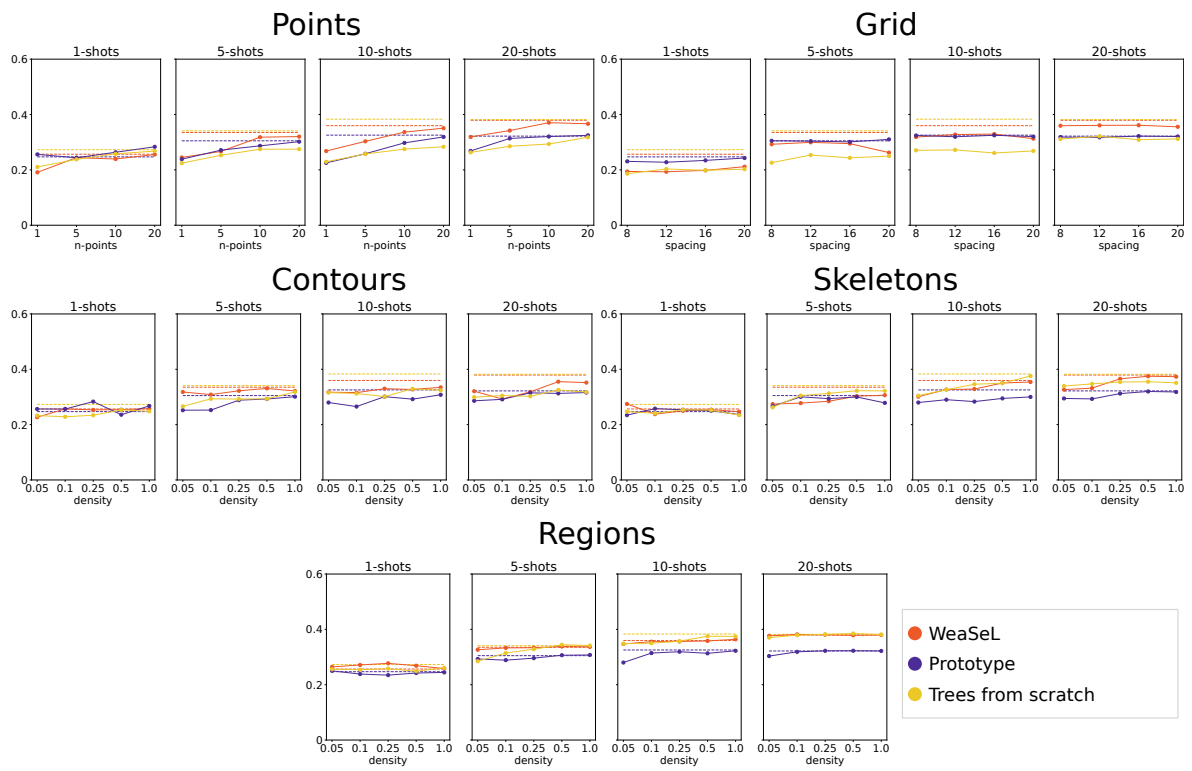


Figure A.12: Jaccard Score of experiments with ISPRS: Potsdam Tree Task.